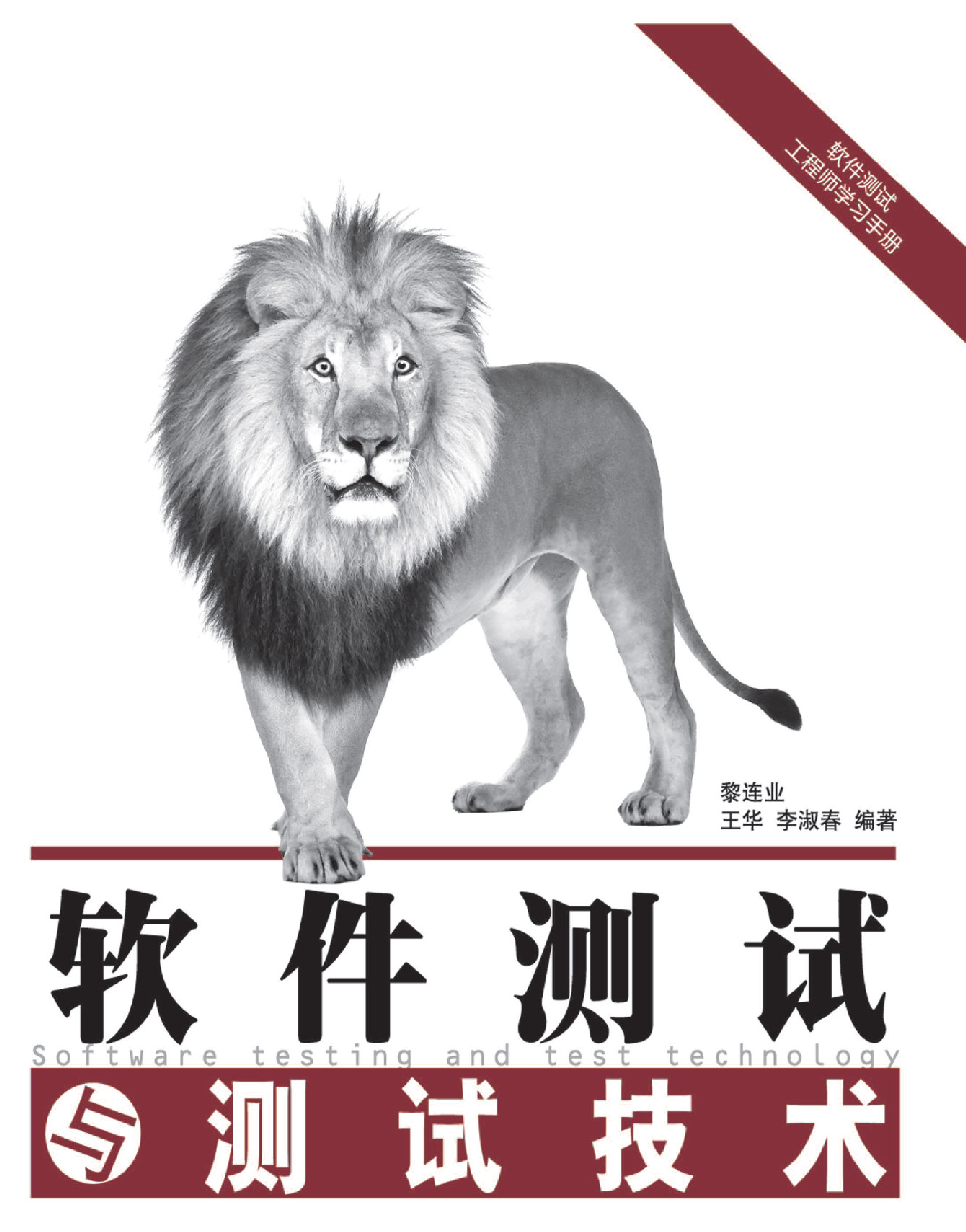


- 来自作者多年的软件工程项目经验和软件信息工程测试监理的工作总结
- 测试用例实战剖析 ◎ 测试文档写作 ◎ 测试项目管理
- 达到独立承担、实施软件测试的能力

清华大学出版社


黎连业
王华 李淑春 编著
软件测试
Software testing and test technology
与测试技术

清华大学出版社
北 京

内 容 简 介

本书根据作者长年项目开发与工程验收等丰富的实践经验，以一个现实的电子政务基础平台系统的案例为线索，重点讲解了软件测试的理论、实践、管理知识，深入剖析和探讨了各种测试类型和不同阶段比较成熟的技术以及测试方法；全书包括单元测试、功能测试、网络测试、性能测试、集成测试、系统测试、验收测试、We 测试、自动化测试、面向对象测试等内容。

本书面向软件测试的实际应用，从组建测试队伍，剖析各阶段的测试内容到通过实例讲解测试用例的组织、设计以及测试文档的编写、测试项目的管理。内容化繁为简，将抽象理论知识转化为可触摸的实际操作，达到更好地理解和消化理论的目的。

全书内容实用，讲述浅显易懂，既可作为高等院校软件测试课程的教材，也可作为软件测试人员的自学用书。对于那些希望强化软件测试技术的程序员、软件项目经理和软件开发团队的相关人员，本书也具有很好的参考价值。

图书在版编目（CIP）数据

软件测试与测试技术 / 黎连业，王华，李淑春编著.－北京：清华大学出版社，2009.5

ISBN978-7-302-19873-4

I.软…　II.①黎…②王…　III.软件－测试　Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字（2009）第 051066 号

责任编辑：夏非彼　张　楠
装帧设计：图格新知
责任校对：闫秀华
责任印制：
出版发行：清华大学出版社　　**地　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编：**100084
社 总 机：010-62770175　　**邮　购：**010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　销：全国新华书店
开　本：190×260　　**印　张：**24.5　　**字　数：**627 千字
附光盘 1 张
版　次：　　**印　次：**
印　数：
定　价：39.00

本书如存在文字不清、漏印以及缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：（010）62770177 转 3103　产品编号：

前言

Preface

本书分别从理论、实践、管理的角度介绍了当前软件测试行业所使用的技术，内容基于软件测试理论和软件测试技术展开，覆盖面广，基本上反映了当前软件测试行业所用的所有技术，是编著者长期从事软件信息工程测试监理的经验总结。

软件测试人员不仅需要掌握软件测试的基本概念和测试技术，还需要具备以下能力：编制测试大纲、测试计划，设计测试用例，撰写测试文档：因为后者是独立承担、实施项目测试的基础，本书对此也结合案例进行了详细全面的描述。

图书内容

本书在内容上可划分为软件测试理论、软件测试实践、测试文档编写、软件测试管理 4 大部分。

在软件测试理论部分，介绍了软件测试的基础知识、软件测试的质量要求、白盒测试技术、黑盒测试技术、单元测试技术、功能测试技术、网络测试和软件安装测试技术、性能测试技术、集成测试技术、系统测试技术、验收测试技术等。

在软件测试实践部分，介绍了软件缺陷测试、测试评估及测试用例设计，并提供了如下实例：

- 界面测试用例设计实例。
- 登录、添加、删除、查询模块测试用例设计实例。
- 宽带接入网网络管理系统测试用例设计实例。
- 某部电子政务应用平台测试用例设计实例。
- 电子政务应用平台主页功能测试用例设计实例。

在测试文档编写部分介绍了 11 类测试文档的写作内容和写作方法：测试需求说明书、测试任务说明书、测试计划说明书、测试大纲写作、测试用例写作、测试分析报告、程序错误报告、集成测试报告、单元测试报告、系统测试报告、验收测试报告。

在软件测试管理部分介绍了测试项目管理、测试过程管理、组织和人员管理、软件配置管理、软件缺陷管理、变更请求管理、进度管理、风险管理、成本管理等。

学习目标

通过本书的学习，读者可以掌握软件测试的基本概念；掌握软件测试的相关技术、工具、方法；掌握关键实施技巧的技术、方法：具有独立承担、实施测试项目的能力。

图书特点

- 内容系统全面、重点突出。
- 叙述由浅入深、循序渐进。
- 概念清楚易懂、案例实用性强。
- 案例、文档模板拿来就用。

目标读者

- 软件测试技术人员。
- 高等院校软件工程专业师生。
- 软件工程专业的技术人员。
- 软件质量管理人员。
- 软件监理人员。

本书是在软件信息工程监理的基础上，参考了大量的技术资料、书籍、文章，并且引用了部分书籍、文章里的图表编写而成的，同时书中还饱含与同行交流的心得体会，写作过程中也得到了许多热心朋友的支持和帮助，在此对所有帮助、鼓励过本人完成此书的朋友表示衷心的感谢；同时由于图书篇幅所限，引用的文献名称和作者就不再一一列出，借此机会一并表示感谢!

本书由黎连业、王华、李淑春编写，并且张宜、黎长骏、张维、单银根、陈建华、王月冬、黎娜、黎军等同志也参与了部分章节的编写工作并提出了许多有益的建议，同时王安、金陆.、段兆金等同志为本书的写作提供了许多技术资料。

由于作者水平有限，书中难免存在疏漏和错误之处，恳请专家和广大读者批评指正。在学习过程中，遇到疑难问题，可以通过以下方式与我们联系：booksaga@126．com，也可以登录图格新知网站 http：／／www,booksaga．com 留言，我们将在第一时间给予答复!

黎连业
2009．04

目录

Contents

第 1 章　软件测试概述 1

1.1　软件的基础知识概述......1
1.1.1　软件的概念、特点和分类......1
1.1.2　软件工程的定义、内容、目标、问题......3
1.1.3　软件生存周期及其模型......5
1.1.4　软件开发方法......12
1.1.5　软件生存周期过程......15
1.1.6　软件工程标准......18
1.1.7　软件开发文档......20
1.1.8　软件质量保证......20
1.2　软件测试的概念、方法和任务......26
1.2.1　软件测试的概念......26
1.2.2　软件测试的方法......34
1.2.3　软件测试的任务......36
1.3　软件测试的术语定义......36
1.4　软件测试的人员要求......44
1.4.1　系统测试人员的结构......44
1.4.2　软件测试人员需要的知识......45
1.4.3　软件测试人员需要的素质......45
1.4.4　软件测试人员的职责......46
1.5　软件测试的前景......46

第 2 章　软件测试的质量要求 48

2.1　软件测试的成熟度模型......48
2.2　软件测试的流程图......55
2.3　软件测试的流程细则......58

第 3 章　白盒测试技术 61

3.1　白盒测试的基本概念......61

3.2 白盒测试的依据和流程......62
3.3 白盒测试的方法......63
3.3.1 代码检查法......63
3.3.2 静态结构分析法......64
3.3.3 静态质量度量法......65
3.3.4 逻辑覆盖法......65
3.3.5 基本路径测试法......68
3.3.6 域测试法......70
3.3.7 符号测试法......70
3.3.8 Z 路径覆盖法......70
3.3.9 程序变异测试法......70
3.4 白盒测试的要求......71
3.4.1 软件各层公用问题测试的要求......71
3.4.2 Java 语言测试的要求......75
3.4.3 数据类型测试的要求......75
3.4.4 SQL 语句测试的要求......75
3.4.5 界面测试的要求......80
3.4.6 数值对象测试的要求......82
3.4.7 业务对象测试的要求......82
3.4.8 数据管理对象测试的要求......83
3.5 白盒测试的工具......83
3.5.1 代码测试工具......83
3.5.2 静态测试和静态测试工具......87
3.5.3 动态测试和动态测试工具......92

第 4 章 黑盒测试技术 95

4.1 黑盒测试的基本概念......95
4.1.1 黑盒测试的优点和缺点......96
4.1.2 黑盒测试与白盒测试的比较......96
4.2 黑盒测试的方法......97
4.2.1 等价类划分方法......97
4.2.2 边界值分析方法......100
4.2.3 错误推测方法......101
4.2.4 判定表驱动分析方法......101
4.2.5 因果图方法......102
4.2.6 正交实验设计方法......107
4.2.7 功能图分析方法......108
4.2.8 场景设计方法......108
4.3 黑盒测试的工具......109

4.3.1 QACenter 测试工具......109
4.3.2 WinRunner 测试工具......111
4.4 黑盒测试的操作步骤......112

第 5 章 软件测试模型和测试工作指南 113

5.1 软件测试工作概述......113
5.1.1 软件测试工作流程......113
5.1.2 软件测试阶段......114
5.2 软件测试模型......115
5.2.1 V 模型......115
5.2.2 W 模型......116
5.2.3 H 模型......117
5.2.4 X 模型......117
5.3 软件测试工作指南......117

第 6 章 单元测试技术 119

6.1 单元测试的内容......119
6.2 单元测试的要点剖析......122

第 7 章 功能测试技术 124

7.1 功能测试概述......124
7.2 功能测试的流程......126
7.3 功能测试用例的书写内容......128

第 8 章 网络测试和软件安装测试技术 130

8.1 网络产品的测试......130
8.1.1 防火墙产品测试......131
8.1.2 入侵检测产品测试......133
8.1.3 入侵防护测试......136
8.1.4 漏洞扫描测试......137
8.1.5 防病毒测试......138
8.1.6 交换机测试......140
8.1.7 服务器测试......142
8.2 网络本身的测试......143
8.2.1 网络测试的类型......143
8.2.2 网络测试的内容......146
8.2.3 网络测试的方式......147
8.2.4 网络应用系统的测试......147
8.2.5 网络性能测试的环境......150

8.2.6 网络应用系统的测试阶段划分......150
8.2.7 网络应用系统的主要测试设备......151
8.3 软件安装的测试......152
8.3.1 共享软件安装测试......152
8.3.2 用户应用系统软件安装测试......153

第 9 章 性能测试技术 155

9.1 性能测试概述......155
9.1.1 性能测试的分类......155
9.1.2 性能测试的目的......159
9.1.3 性能测试的指标......159
9.1.4 性能测试的内容......159
9.1.5 性能测试的策略......159
9.1.6 性能测试的方法......160
9.2 性能测试的实例剖析......161
9.2.1 并发性能测试剖析......161
9.2.2 Web 站点质量分析剖析......162
9.2.3 应用故障定位剖析......163
9.2.4 测试策略剖析......163

第 10 章 集成测试技术 167

10.1 集成测试概述......167
10.1.1 集成测试过程......167
10.1.2 集成测试方法......168
10.2 集成测试阶段工作......171

第 11 章 系统测试技术174

11.1 系统测试的主要内容和测试类型......174
11.2 系统测试的过程......175
11.3 系统测试的结果分析......176
11.4 系统测试的文档资料......178

第 12 章 验收测试技术 180

12.1 验收测试的先决条件......180
12.2 验收测试的目的......180
12.3 验收测试的内容......181

第 13 章 Web 测试技术 182

13.1 Web 的功能测试......182

13.2 Web 的性能测试......184
13.3 Web 的用户界面测试......185
13.4 Web 的兼容性测试......187
13.5 Web 的安全性测试......188
13.6 Web 的接口测试......188

第 14 章 自动化测试技术 190

14.1 自动化测试概述......190
14.2 自动化测试技术......193
14.3 自动化测试级别......195
14.4 自动化测试框架......196
14.5 自动化测试工具......198
14.5.1 自动化测试工具的特征......198
14.5.2 自动化测试工具的分类......198
14.5.3 自动化测试工具的常用类型......199

第 15 章 面向对象的测试技术 204

15.1 面向对象的测试概述......204
15.1.1 面向对象的基本概念......204
15.1.2 类的特性......205
15.1.3 面向对象的开发方法......207
15.1.4 面向对象的模型......210
15.1.5 面向对象的设计......212
15.1.6 面向对象的测试内容......213
15.1.7 面向对象的测试模型......215
15.2 面向对象分析的测试......217
15.3 面向对象编程的测试......221
15.4 面向对象的单元测试......222
15.4.1 类的测试和测试要求......222
15.4.2 类测试设计的方法......222
15.4.3 单元测试使用的方法......224
15.5 面向对象的集成测试......225
15.5.1 面向对象集成测试的目的......225
15.5.2 面向对象集成测试的策略......226
15.5.3 面向对象集成测试的静态和动态测试......226
15.5.4 面向对象集成测试的用例和测试过程......227
15.5.5 面向对象集成测试的常见故障......229
15.6 面向对象的系统测试......230
15.7 面向对象软件的测试用例设计......230

15.7.1 面向对象软件的测试用例设计原则......230
15.7.2 面向对象软件的测试用例设计方法......231

第 16 章 软件缺陷测试和测试评估 237

16.1 软件缺陷概述......237
16.1.1 软件缺陷的定义......237
16.1.2 软件缺陷的特征......238
16.1.3 软件缺陷的类型......238
16.1.4 Bug 状态......239
16.1.5 Bug 的等级划分与优先级......239
16.1.6 软件缺陷的标识、种类和属性......240
16.1.7 缺陷的起源、来源和根源......241
16.1.8 Bug 记录......242
16.2 软件缺陷的生命周期......243
16.3 软件缺陷的跟踪管理......244
16.3.1 软件缺陷的测试报告......244
16.3.2 软件缺陷的分离和重现......246
16.3.3 软件缺陷的跟踪系统......247
16.4 软件测试的评估......248
16.4.1 测试覆盖评估......248
16.4.2 软件测试的质量评估......250
16.4.3 软件测试的缺陷评估......253
16.4.4 软件测试的性能评估......255

第 17 章 测试用例设计和电子政务应用平台测试用例设计实例 257

17.1 测试用例的基本概念......257
17.1.1 测试用例概述......257
17.1.2 测试用例设计......259
17.2 界面测试用例设计实例......261
17.3 登录、添加、删除、查询模块测试用例设计实例......272
17.4 宽带接入网网络管理系统测试用例设计实例......274
17.5 某部电子政务应用平台测试用例设计实例......279
17.5.1 主页信息发布测试用例设计实例......279
17.5.2 工作站设置测试用例设计实例......280
17.5.3 文件维护测试用例设计实例......281
17.5.4 查询显示页面测试用例设计实例......282
17.5.5 数据传输测试用例设计实例......282
17.5.6 个人信息通信工具测试用例设计实例......283
17.5.7 公文管理测试用例设计实例......283

17.5.8 修改文件和修改撰文单位测试用例设计实例......287
17.5.9 党、团、工会事务管理测试用例设计实例......288
17.5.10 贺电事务管理测试用例设计实例......288
17.5.11 固定资产管理测试用例设计实例......289
17.5.12 会务管理测试用例设计实例......290
17.5.13 领导日程管理测试用例设计实例......291
17.5.14 机构管理维护测试用例设计实例......291
17.5.15 代码维护和主题词分类测试用例设计实例......292
17.5.16 公文流转测试用例设计实例......293
17.5.17 目录测试用例设计实例......293
17.5.18 维护人员测试用例设计实例......294
17.6 电子政务应用平台主页功能测试用例设计实例......294

第 18 章 测试文档的写作 296

18.1 测试文档的写作概述......296
18.2 测试需求说明书写作的内容......298
18.2.1 测试需求说明书的写作方法......298
18.2.2 测试需求说明书的写作模板......298
18.3 测试任务说明书写作的内容......302
18.3.1 测试任务、测试质量和测试范围......302
18.3.2 确定测试进度和管理......303
18.3.3 测试注意事项......304
18.4 测试计划说明书写作的内容......304
18.5 测试大纲写作的内容......307
18.6 测试用例写作的内容......311
18.7 测试分析报告写作的内容......317
18.7.1 测试分析报告模板的目录......317
18.7.2 测试分析报告模板的写作内容......318
18.8 集成测试报告写作的内容......327
18.9 单元测试报告写作的内容......331
18.10 系统测试总结报告写作的内容......333
18.10.1 系统测试总结报告模板的图示......333
18.10.2 系统测试总结报告模板的写作要点......334
18.11 验收测试报告写作的内容......336

第 19 章 软件的其他测试技术 344

19.1 可用性测试......344
19.2 安全性测试......344
19.3 强度测试或压力测试......345

19.4 确认测试......345
19.5 容错性测试......346
19.6 回归测试技术......346
19.7 易用性测试......348

第 20 章 软件测试管理 349

20.1 测试管理概述......349
20.2 测试项目管理......350
20.3 测试过程管理......353
20.4 组织和人员管理......355
20.4.1 软件测试的组织......355
20.4.2 软件测试组织的职能......357
20.4.3 软件测试的组织结构......357
20.4.4 软件测试组织结构的准则......357
20.4.5 软件测试人员的能力要求......357
20.5 软件配置管理......358
20.5.1 软件配置管理概述......358
20.5.2 软件配置管理要求......364
20.6 软件缺陷管理......366
20.7 变更请求管理......367
20.8 进度管理......368
20.9 风险管理......369
20.9.1 软件风险的基本概念......369
20.9.2 风险识别和分析......372
20.9.3 软件项目风险管理模型......374
20.10 成本管理......374

第1章　软件测试概述

软件测试是信息系统开发中不可缺少的一个重要步骤，随着软件变得日益复杂，软件测试也变得越来越重要。软件的基础知识、软件测试的概念（方法、目标和任务）、软件测试的术语定义是软件测试的基础。本章重点讨论以下内容：

- 软件的基础知识概述。
- 软件测试的概念、方法和任务。
- 软件测试的术语定义。
- 软件测试的人员要求。
- 软件测试的前景。

1.1　软件的基础知识概述

1.1.1　软件的概念、特点和分类

1. 软件的概念

软件是计算机系统中与硬件相互依存的一部分，包括程序、数据以及与其相关文档的完整集合。其中，程序是按事先设计的功能和性能要求执行的指令序列；数据是使程序能正常操纵信息的数据结构；文档是与程序开发、维护和使用有关的图文材料。

2. 软件的特点

软件具有 8 个特点：

- 软件是一种逻辑实体，而不是具体的物理实体，因而它具有抽象性。
- 软件的生产与硬件不同，它没有明显的制造过程。对软件的质量控制，必须着重在软件开发方面下功夫。
- 在软件的运行和使用期间，没有硬件那样的机械磨损和老化问题，然而它存在退化问题，必须要对其进行多次的修改与维护。
- 软件的开发和运行常常受到计算机系统的制约，对计算机系统有着不同程度的依赖性。为了解除这种依赖性，在软件开发中提出了软件移植的问题。
- 软件的开发至今尚未完全摆脱手工艺的开发方式。
- 软件本身是复杂的。软件的复杂性可能来自它所反映的实际问题的复杂性，也可能来自程序逻辑结构的复杂性。

- 软件成本相当昂贵。软件的研制工作需要投入大量的、复杂的、高强度的脑力劳动，它的成本是比较高的。
- 相当多的软件工作涉及到社会因素。许多软件的开发和运行涉及机构、体制及管理方式等问题，它直接影响到项目的成败。

3. 软件的分类

软件的分类方法有如下 4 种。

（1）按软件的功能分类

按软件的功能分类，软件可以分为系统软件、支撑软件、应用软件。

- 系统软件：是与计算机硬件紧密配合在一起，使计算机系统各个部件、相关的软件和数据协调、高效地工作的软件，例如操作系统、数据库管理系统、设备驱动程序以及通信处理程序等。
- 支撑软件：是协助用户开发软件的工具性软件，其中包括帮助程序人员开发软件产品的工具，也包括帮助管理人员控制开发进程的工具。
- 应用软件：是在特定领域内开发、为特定目的服务的一类软件，其中包括为特定目的进行的数据采集、加工、存储和分析服务的资源管理软件。

（2）按软件服务对象的范围分类

按软件服务对象的范围分类，软件可以分为项目软件和产品软件。

- 项目软件：也称定制软件，是受某个特定客户（或少数客户）的委托，由一个或多个软件开发机构在合同的约束下开发出来的软件，例如军用防空指挥系统、卫星控制系统。
- 产品软件：是由软件开发机构开发出来直接提供给市场，或是为千百个用户服务的软件，例如文字处理软件、财务处理软件、人事管理软件等。

（3）按软件规模分类

按开发软件所需要的人力、时间以及完成的源程序行数，可确定 6 种不同规模的软件，如表 1-1 所示。

表 1-1　按软件规模分类

类别	参加人员数	研制期限	产品规模（源程序行数）
微型	1	1～4 周	500 行
小型	1	1～6 月	1000～2000 行
中型	2～5	1～2 年	5000～50000 行
大型	5～20	2～3 年	50000～100000 行
甚大型	100～1000	4～5 年	1000000 行
极大型	2000～5000	5～10 年	1000000～10000000 行



不论规模大、时间长、很多人参加的软件项目，还是规模小、时间短、参加人员少的软件项目，其开发工作必须要有软件工程的知识作为指导，遵循一定的开发规范，其基本原则是一样的，只是对软件工程技术依赖的程度不同而已。

（4）按软件的工作方式分类

按软件的工作方式分类，软件可以分为实时处理软件、分时软件、交互式软件、批处理软件。

- 实时处理软件：指在事件或数据产生时立即予以处理，并及时反馈信号，控制需要监测和控制的过程的软件，主要包括数据采集、分析、输出三部分。
- 分时软件：允许多个联机用户同时使用计算机。
- 交互式软件：能实现人机通信的软件。
- 批处理软件：把一组输入作业或一批数据以成批处理的方式一次运行，按顺序逐个处理完的软件。

1.1.2 软件工程的定义、内容、目标、问题

1. 软件工程的定义

软件工程有多种定义，其中鲍姆（B.W.Boehm）曾为软件工程下的定义是：运用现代科学技术知识来设计且构造计算机程序，并且包括为开发、运行和维护这些程序所必需的相关文件资料。

这个定义说明了软件工程是计算机科学中的一个分支，其主要思想是在软件生产中用工程化的方法代替传统手工方法。工程化的方法借用了传统的工程设计原理的基本思想，采用若干科学的、现代化的方法技术来开发软件，这种工程化的思想贯穿到需求分析、设计、实现直到维护的整个过程。

1983 年 IEEE 给出的定义是：软件工程是开发、运行、维护和修复软件的系统方法。软件工程具有如下的性质：

- 软件工程是一门综合性的交叉学科，它涉及计算机科学、工程科学、管理科学、数学等领域。计算机科学中的研究成果均可用于软件工程，但计算机科学着重于原理和理论，而软件工程着重于如何建立建造一个软件系统。
- 软件工程要用工程科学中的观点来进行费用估算、制定进度、制定计划和方案。
- 软件工程要用管理科学中的方法和原理进行软件生产的管理。
- 软件工程要用数学的方法建立软件开发中的各种模型和各种算法，如可靠性模型、说明用户需求的形式化模型等。

2. 软件工程的内容

软件工程研究的主要内容是软件开发技术和软件开发管理两个方面，其中在软件开发技术中，主要研究软件工程方法、软件工程过程、软件开发工具和环境。

- 软件工程方法为软件开发提供了“如何做”的技术，它包括多方面的任务，如项目计划与估算、软件系统需求分析、数据结构、系统总体结构的设计、算法的设计、编码、测

试以及维护等。软件工程方法常要采用某种特殊的语言或图形的表达方法以及一套质量保证标准。

- 软件工程过程是将软件工程的方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的。过程定义了方法使用的顺序、要求交付的文档资料，为保证质量和协调变更所需要的管理。
- 软件开发工具和环境为软件工程方法提供了自动的或半自动的软件支撑环境。目前，已经开发出了许多软件工具，从而能够支持上述的软件工程方法，而且已经有人把诸多软件工具集成起来，使得一种工具产生的信息可以为其他的工具所使用，这样建立起一种称之为计算机辅助软件工程（CASE）的软件开发支撑系统。CASE 将各种软件工具、开发机器和一个存放开发过程信息的工程数据库组合起来形成一个软件工程环境。

3. 软件工程的目标

软件工程是一门工程性学科，目的是成功地建造一个大型软件系统，所谓成功是要达到以下几个目标：

- 付出较低的开发成本。
- 达到要求的软件功能。
- 取得较好的软件性能。
- 开发的软件易于移植。
- 需要较低的维护费用。
- 能按时完成开发任务，及时交付使用。
- 开发的软件可靠性高。

在实际开发的具体项目中，要想让以上几个目标都达到理想的程度往往是非常困难的，而且上述目标很可能是互相冲突的，若只降低开发成本，很可能同时也降低了软件的可靠性。另一方面，如果过于追求提高软件的性能，可能造成开发出的软件对硬件有较大的依赖，从而直接影响到软件的可移植性。

软件工程的目标之间的相互关系如图 1-1 所示。

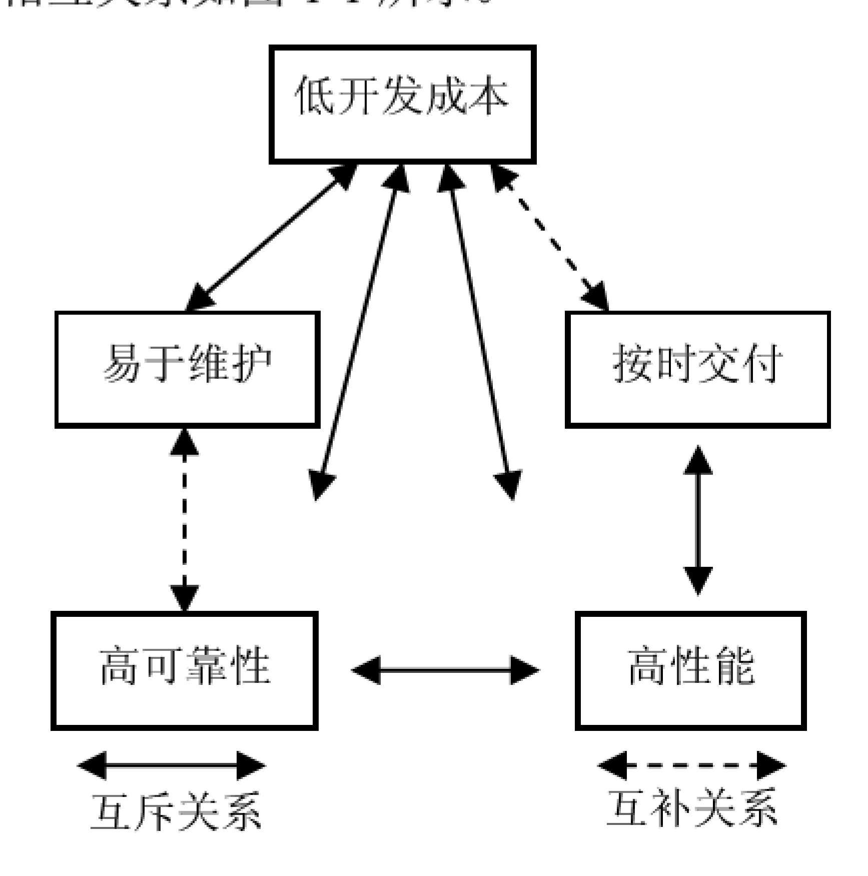


图 1-1　软件工程的目标之间的相互关系

其中：

- 易于维护和高可靠性之间、低开发成本与按时交付之间为互补关系。
- 低开发成本和易于维护、低开发成本和高可靠性、低开发成本和高性能、高可靠性和高性能、高性能和按时交付之间为互斥关系。

4. 软件工程的问题

软件工程面临的问题主要有软件费用、软件可靠性、软件可维护性、软件生产率和软件重用等。

（1）软件费用

软件是知识高度密集的技术的综合产物，软件人力资源不能适应软件迅速增长的社会要求，因此，软件费用上升是必然趋势。

（2）软件可靠性

软件可靠性是指软件系统能否在既定的环境条件下运行并实现所期望的结果。在软件开发中，通常要花费 40%的代价进行测试和排错，为了提高软件可靠性，就要付出足够的代价。

（3）软件可维护性

软件维护费用占整个软件系统费用的 2/3，而软件开发费用只占整个软件系统费用的 1/3，这是因为已经运行的软件还需要排除隐含的错误，新增加的功能要加入进去，维护工作是非常困难的，效率较低，因此，如何提高软件的可维护性、减少软件维护的工作量也是软件工程面临的主要问题之一。

（4）软件生产率

计算机的广泛应用使得软件的需求量大幅度上升，而软件的生产又处于手工开发的状态，软件生产率低下，使得软件开发人员严重不足，所以，如何提高软件生产率是软件工程又一重要问题。

（5）软件重用

提高软件的重用性，对于提高软件生产率、降低软件成本有重要意义。当前的软件开发存在着大量重复的劳动，耗费了不少的人力资源。软件重用有各种级别，软件规格说明、软件模块、软件代码、软件文档等都可以是软件重用的单位。软件重用是软件工程中的一个重要研究课题。

1.1.3　软件生存周期及其模型

1. 软件生存周期

软件生存周期是指一个软件从提出开发要求开始直到该软件报废为止的整个时期，软件生存周期划分为 6 个阶段，即制定计划、需求分析、软件设计、程序编码、软件测试及运行维护。

（1）制定计划

制定计划的目的是确定要开发软件系统的总目标，给出它的功能、性能、可靠性以及接口等

方面的要求；由系统分析员和用户合作，研究完成该项软件任务的可靠性，探讨解决问题的可能方案，并对可利用的资源（计算机硬件、软件、人力等）、成本、可取得的效益、开发的进度做出估计，制定出完成开发任务的实施计划，连同可行性研究报告提交到管理部门审查。

（2）需求分析

需求分析阶段的任务不是具体地解决问题，而是准确地确定软件系统必须做什么，确定软件系统必须具备哪些功能，软件开发人员要和用户密切配合，充分交流各自的想法、充分理解用户的业务流程，完整、全面地收集、分析用户信息，从中分析出用户要求的功能和性能，并对其加以确切地描述，然后编写出软件需求说明书或系统功能说明书，并提交到管理机构评审。

（3）软件设计

设计是软件工程的技术核心。设计可分为概要设计和详细设计两个阶段。

在概要设计阶段，开发人员要把确定的各项功能需求转换成需要的体系结构，在该体系结构中，每个成份都是意义明确的模块，即每个模块都和某些功能需求相对应。概要设计的主要任务如下：

- 设计软件结构。
- 明确该结构由哪些模块组成。
- 确定模块的层次结构和调用关系。
- 设计系统的总体数据结构和数据库结构。

详细设计阶段就是为每个模块完成的功能进行具体描述，要把功能描述转变为精确的、结构化的过程描述，即该模块的控制结构是怎样的、应该先做什么、后做什么、有什么样的条件判定、有什么重复处理等，并用相应的表示工具把这些控制结构表示出来。

（4）程序编码

程序编码就是把每个模块的控制结构转换成计算机可以接受的程序代码，即写成从某一种特定程序设计语言表示的“源程序清单”。写出的程序应结构好、清晰易读，并且与设计相一致。

（5）软件测试

软件测试是保证软件质量的重要手段，其主要方式是在设计测试用例的基础上检验软件的各个组成部分。测试分为单元测试、集成测试、有效性测试等。单元测试是查找各模块在功能和结构上存在的问题。集成测试是将各模块按一定顺序组装起来进行测试，主要是查找各模块之间接口上存在的问题。有效性测试是按需求说明书上的功能逐项测试，决定开发的软件是否合格、能否交付用户使用等。

（6）运行维护

软件投入正式使用后，便进入运行与维护阶段。软件在运行过程中可能由于各方面的原因，需要对它进行修改。其原因可能是：运行中发现了软件隐含的错误而且需要修改；为了适应变化了的软件工作环境而需要进行适当变更；为了增强软件的功能需要进行变更等。

2. 软件生存周期模型

软件生存周期模型是描述软件开发过程中各种活动如何执行的模型。它确立了软件开发中各阶段的次序限制以及各阶段活动的准则，确立了开发过程中需要遵守的规定和限制，便于各种活动的协调、各种人员的有效通信，有利于活动重用和活动管理。

软件生存周期模型主要有瀑布模型、原型法模型、螺旋模型、喷泉模型、智能模型。

（1）瀑布模型

瀑布模型（LCA Life Circle Approach）又称生命周期法，在 20 世纪 90 年代以前，系统设计主要是使用瀑布模型。瀑布模型的理论认为：任何一个软件都有它的生存期。所谓软件的生存期是指从软件项目的提出后，经历研制、运行和维护，直至退出的整个时期。瀑布模型将信息系统的整个生存期视为一个生命周期，同时又将整个生存期严格划分为若干阶段，并明确每一阶段的任务、原则、方法、工具和形成的文档资料，分阶段、分步骤地进行信息系统的设计。生命周期法将信息系统的生命周期划分为系统规划、系统分析、系统设计、系统实施、系统运行维护与评价共 5 个阶段，各个阶段完成的主要任务如下。

- 系统规划阶段：系统规划主要是由系统分析员和用户讨论，通过对现行系统的调查确定管理信息系统的目标及总体功能结构、规划管理信息系统开发的费用及进度、从整体上研究企业管理（或业务）流程的现状及存在的问题、按照用户的资金和技术力量分析信息系统是否可行，并将分析结果以可行性报告的形式提交给相关决策人。
- 系统分析阶段：系统分析的任务是在对现有信息系统进行详细调研的基础上．通过各种可能的方式充分描述现有系统的业务流程及所需处理的数据，并分析这些处理过程及数据结构的逻辑合理性，从而构思和确立新系统的基本目标和逻辑功能，给出新系统的逻辑方案，最后写出系统分析说明书，即系统的总体设计方案。
- 系统设计阶段：系统设计的任务是依据系统分析所得到的系统功能和信息需求设计系统的处理流程及相关数据类型，确定系统的应用软件结构，包括对处理系统的模块设计、代码设计、数据文件设计、输入输出设计、处理逻辑设计等，从而确定信息系统的物理模型，最后形成系统设计说明书。
- 系统实施阶段：系统实施的主要任务包括硬件设备的购置、安装，依据系统设计的要求完成每一应用模块的程序设计、组装调试、系统测试、系统切换、操作人员的培训等工作。
- 系统运行维护与评价阶段：系统运行维护与评价阶段的主要工作包括系统运行、维护、运行管理相应新系统，并从目标、功能、性能及经济效益方面对系统进行评价。

瀑布模型具有以下特点：

- 采用结构化思想，其开发策略是“自顶向下”地完成管理信息系统的规划、分析与设计工作，然后“自底向上”地实现。
- 开发过程阶段清楚、任务明确、文档齐全，并要求有标准化的分析报告和文本等阶段性文档资料及书面审定记录，使得整个开发过程便于管理和控制。
- 通常假定系统的应用需求是预先描述清楚的，排除了不确定性。
- 瀑布模型适用于大型的信息系统以及应用软件的开发。

瀑布模型的优点是：在消除非结构化软件、降低软件的复杂性、促进软件开发工程化方面起了很大作用。

虽然瀑布模型的理论比较完善，但也存在一些缺陷，主要表现在以下几个方面。

- 系统开发周期长、见效慢：在系统实施的前几个阶段能给用户提供的只是文字上的方案，用户长期看不到可运行的系统，不能在短期内得到效益，积极性受挫，导致其下不了组建系统的决心。即使下了开始组建系统的决心，但由于在开发过程中，用户一直没有有效途径与开发人员共同研究系统分析之后的阶段，在这几个阶段因为需要用到专门的繁琐的图表工具，使得系统开发过程处于“闭门造车”的封闭状态，管理者基本无法参与其中。由于生命周期法严格依据各阶段的目标和任务进行开发，使得开发周期拖长，—个规模较大的系统开发过程往往需要三到五年，这样一方面使用户在较长时间内不能得到一个可实际运行的物理系统；另一方面是在开发周期内计算机理论和技术会不断发展与更新、用户的组织结构及信息需求也会发生变化，系统尚未开发出来可能就已经过时了。
- 难沟通、效率低：该方法要求系统设计者在系统开发之初全面认识并明确表达系统的信息需求，充分预料各种可能发生的变化，然而这是十分不现实的。往往许多系统的建设是在开发过程中逐步明确和完善的，特别对于侧重于辅助决策的管理信息系统的开发更是如此。在系统调查阶段，由于专业及知识背景不一样，各级管理人员往往自己也不能确切地描绘未来管理信息系统的目标，分析人员在理解上也会有误解和偏差，造成系统需求定义的困难，另外又要求系统对不断变化的内外部环境具有一定的适应性，然而这正是瀑布模型所不允许的。
- 开发过程灵活性小、更改成本高：在瀑布模型中，系统开发严格按阶段进行，每一阶段的成果一经审核，批准形成文档、资料之后就难于更改，这种做法无法适应今后用户需求的变化，然而稳定是相对的，变化才是必然的。另一方面，系统维护起来仍然相当困难，维护成本也非常高。统计数字表明，信息系统维护的生产率比软件开发的生产率低几十倍。

瀑布模型如图 1-2 所示。

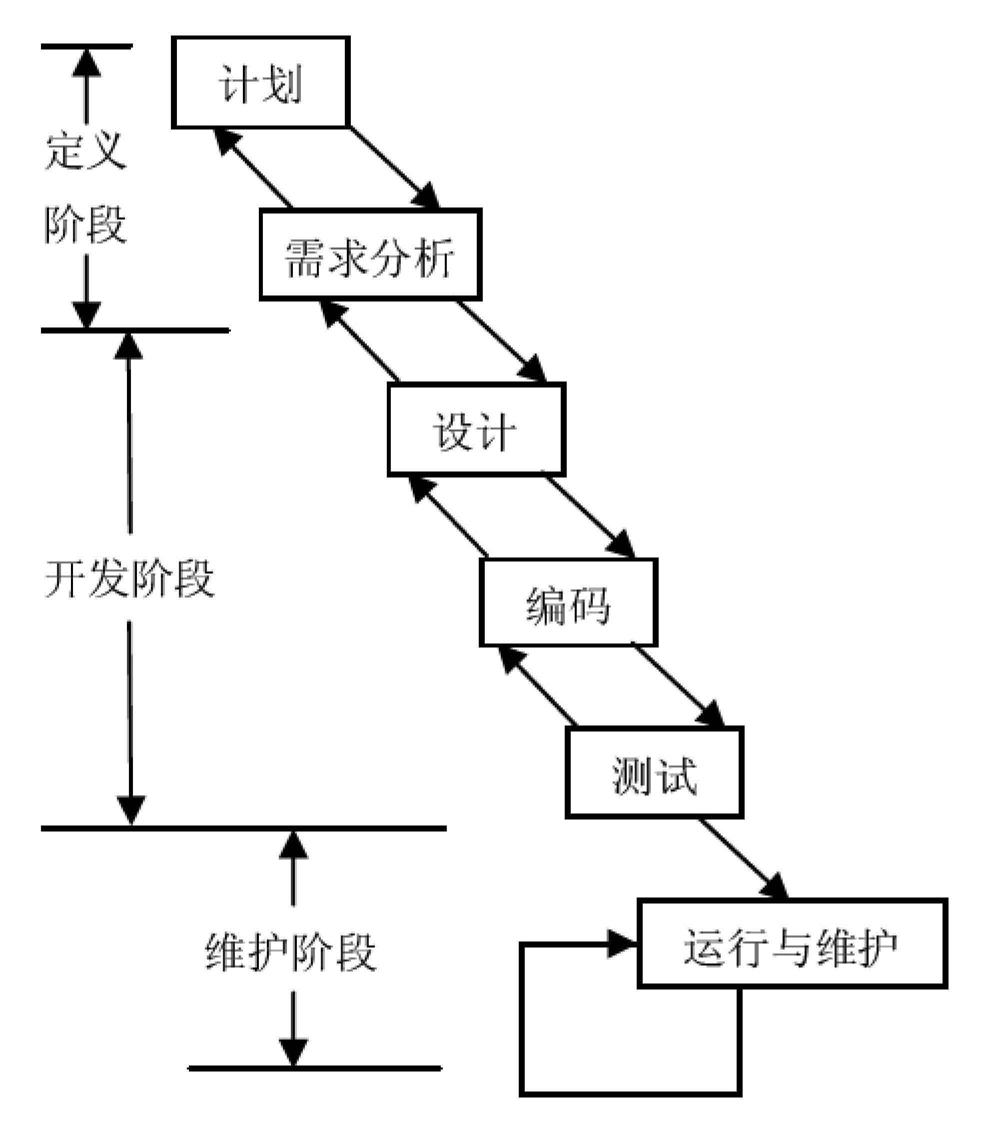


图 1-2　软件生存期的瀑布模型

图 1-3 说明如下几点问题：

- 从上一阶段接受本阶段的工作对象，作为输入。
- 利用这一输入实施本阶段应完成的内容。
- 给出本阶段的工作成果，作为输出传给下一阶段。
- 对本阶段实施的工作进行评审。

瀑布模型为软件开发提供了一种有效的管理模式。根据这一模式制定开发计划、进行成本预算、组织开发力量，以项目的阶段评审和文档控制为手段有效地对整个开发过程进行指导，从而保证了软件产品及时交付，并达到预期的质量要求。瀑布模型将软件生存周期的各项活动规定为依固定顺序连接的阶段工作，是一种线性模型。

瀑布模型的缺点是：缺乏灵活性，特别是无法解决软件需求不明确或不准确的问题。这些问题的存在对软件开发带来了严重影响，最终可能导致开发出的软件并不是用户真正需要的软件。

瀑布模型是一种整体开发模型。在开发过程中，用户对软件的需求认识常常不够清晰，因而使得开发项目难于做到一次开发成功。只有开发完成后，整个软件全部展现在用户面前，这时发现有不满意的地方为时已晚。

（2）原型法模型（演化模型）

原型法也称渐进法或迭代法。原型法要求在获得基本的用户需求后，快速地建立系统的一个“原型”，用户及其他相关人员在试用原型后反馈意见，设计者修改原型后再交给用户试用。通过反复评价和反复修改原型系统，逐步确定各种需求的细节，从而最终完成系统的开发。

实践证明，在信息系统开发初期，用户的需求是经常变动的，有时甚至是十分模糊的。原型法的提出正是为了改善同用户的交流，原型法的含义是要明确用户系统的原始需求，尽量缩短系统开发周期，提高软件开发效率。它克服了生命周期法的一些缺陷。

原型法具有以下特点：

- 用户自始至终参与系统开发全过程，强调了用户的主导作用，这就使实施后系统的切换与运行维护较为容易和自然。
- 用户和管理阶层可以更快地看到可以工作的信息系统原型，从而尽快地发现系统中存在的错误和疏漏，开发效益相应得到了提高。
- 特别适合开发那些需求不确定性较高的信息系统。
- 由于要快速实现新系统的一个原型，因而对开发环境、软件工具要求比较高，需要有合适的软件环境的支持。

虽然原型法克服了生命周期法的一些缺点，但也存在一些缺陷，主要表现在以下几个方面：

- 容易出现系统质量缺陷：原型法鼓励采用了“编码、实现、修复”的开发方法，这样有可能提高整个系统生命周期的运行、支持和维护成本，同时又会失去开发过程中选择更好的技术方案的机会，因为技术人员和用户都希望尽快地看到可以使用的原型，由于这种方法过于强调速度，使得许多潜在的系统质量缺陷没有得到很好地解决。
- 开发过程难以控制：让用户自始至终控制着系统开发的进程将导致开发者不能确定自己的工作进度，使工期无法保证。另外，用户的参与有时可能会成为其自身的一种负担，往往

力不从心，因为，这种参与不是直观地操作，而是处于系统方案与实现系统之间很不直观的论证工作，从而使得开发过程难以控制，项目管理和系统的维护比较困难。

- 不适用于大型系统的开发：对于大型的信息系统工程，由于其复杂性而导致无法快速建立原型，因此原型法比较适合开发小型的、灵活性要求较高的信息系统。对于近代管理者为了应付激烈的竞争需要即时分析优化系统的高要求来说，原型法就无能为力了。

原型法模型又可细分为增量模型、渐进模型和演化模型。

- 增量模型：对于需求不能很快全部明确的系统，软件开发项目难于做到一次开发成功，此时可使用增量模型。应尽可能明确已知的软件需求，完成相应的需求分析，并按瀑布模型的方法进行第一次开发工作。在系统集成时，通过实验找出需求中的欠缺和不足之处，明确那些未知的软件需求，再迭代进行增加部分的需求分析和开发。对于有些系统而言，这种反复可能要进行几次，但尽可能不要超过两次，否则难以控制软件的结构规模、开发质量和进度。
- 渐进模型：此模型主要是针对部分需求尽管明确但一时难以准确进行定义的系统设计，如用户的操作界面等。使用此模型时，可以先做初步的需求分析，之后立即进行设计和编码，随后与系统进行第一次集成（不作或少作测试）。根据集成后反映的问题进一步做更全面的需求分析、设计、编码、测试和集成。
- 演化模型：演化模型是一种非整体开发的模型。软件在该模型中是“逐渐”开发出来的，开发出一部分，向用户展示一部分，可让用户及早看到部分软件、及早发现问题，也可以先开发一个“原型”软件，完成部分主要功能，展示给用户并征求意见，然后逐步完善，最终获得满意的软件产品。演化模型具有较大的灵活性，适合于软件需求不明确、设计方案有一定风险的软件项目。



演化模型和增量模型之间的区别：演化模型首先开发核心系统，每次迭代时都为系统增加一个子集，整个系统是增量开发和增量提交，增量模型首先完整的开发系统的一个初始子集，然后不断建造更精细的版本。

（3）螺旋模型

螺旋模型将瀑布模型与演化模型结合起来，并且加入了两种模型均忽略的风险分析，弥补了这两种模型的不足。

螺旋模型将开发过程分为几个螺旋周期，每个螺旋周期大致和瀑布模型相符合。螺旋模型沿着螺线旋转，即在笛卡尔坐标的 4 个象限上分别表达了 4 个方面的活动，如图 1-3 所示。

图 1-3　螺旋模型

在图 1-4 中可以看出以下几点。

- 制定计划：确定目标、选定实施方案、明确开发限制条件。
- 风险分析：分析所选方案、识别风险、通过原型消除风险。
- 实施工程：实施软件开发。
- 客户评估：评价开发工作、提出修改意见、建立下一个周期的计划。

螺旋模型适合于大型软件的开发，它吸收了软件工程“演化”的概念，使得开发人员和用户对每个螺旋周期出现的风险有所了解，从而作出相应的反应。

螺旋模型的使用有以下两点不足：

- 螺旋模型的使用需要有相当丰富的风险评估经验和专门知识，这使该模型的应用受到一定限制。
- 螺旋模型对软件复用和生存期中多项开发活动的集成并未提供支持，因而难于支持面向对象的开发方法。

（4）喷泉模型

喷泉模型是一种以用户需求为动力、以对象作为驱动的模型，它体现了软件创建所固有的迭代和无间隙特征，喷泉模型主要用于支持面向对象开发过程，适合于面向对象的开发方法。它克服了瀑布模型不支持软件重用和多项开发活动集成的局限性。喷泉模型使开发过程具有迭代性和无间隙性。

- 迭代性：系统某些部分常常重复工作多次，相关功能在每次迭代中随之加入演化的系统。
- 无间隙：指在分析、设计、实现等开发活动之间不存在明显的边界。

喷泉模型的示意如图 1-4 所示。

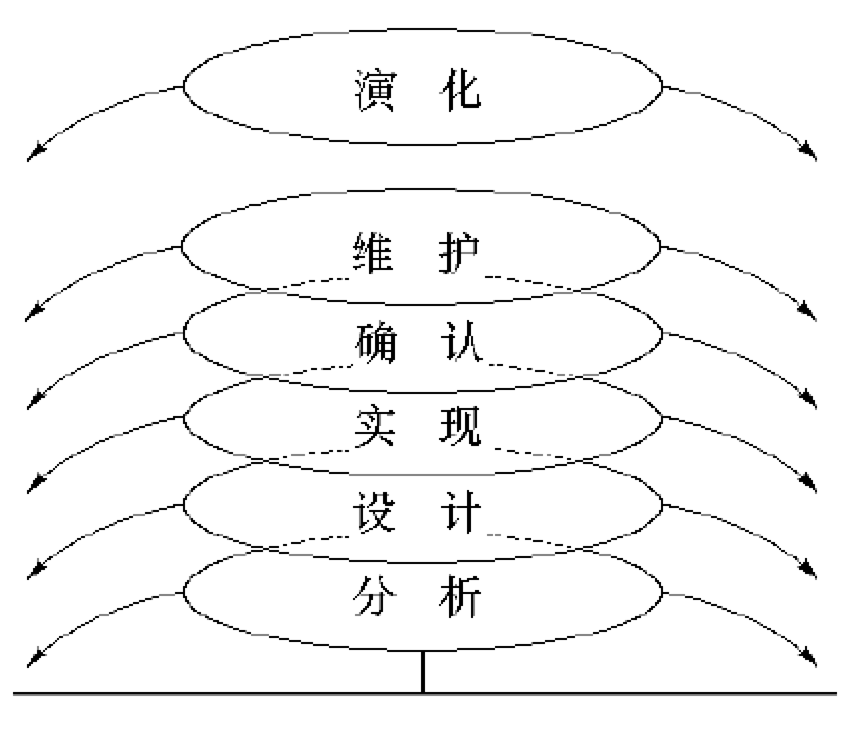


图 1-4　喷泉模型



瀑布模型、演化模型、螺旋模型之间的相同点是这三个模型都分为多个阶段，而瀑布模型一次完成软件，演化模型分为多次完成，每次迭代完成软件的一个部分，螺旋模型也分为多次完成，每次完成软件的一个新原型，并考虑风险分析。

（5）智能模型

智能模型也称为基于知识的软件开发模型，它把瀑布模型和专家系统结合在一起。该模型在开发的各个阶段上都利用相应的专家系统来帮助软件人员完成开发工作，使维护在系统需求说明一级上进行。为此，建立了各阶段所需要的知识库，将模型相应领域的知识、软件工程知识分别存入数据库，以软件工程知识为基础的生成规则构成的专家系统与含有应用领域知识规则的其他专家系统相结合，构成了该领域开发系统。

1.1.4　软件开发方法

软件开发的目标是在规定的投资和时间内，开发出符合用户需求的高质量的软件。为了达到此目的，需要选择成功的软件开发方法。

软件开发方法主要有结构化方法、Jackson 方法、维也纳开发方法、面向对象的开发方法。

1. 结构化方法

结构化方法由结构化分析、结构化设计、结构化程序设计构成，它是一种面向数据流的开发方法。该方法简单实用、应用较广、技术成熟。

- 结构化分析：它是根据分解与抽象的原则，按照系统中数据处理的流程，用数据流图来建立系统的功能模块，从而完成需求分析工作。
- 结构化设计：它是根据模块独立性准则、软件结构准则将数据流图转换为软件的体系结构，用软件结构图来建立系统的物理模型，实现系统的概要设计。
- 结构化程序设计：它是根据结构程序设计原理，将每个模块的功能用相应的标准控制结构表示出来，从而实现详细设计。

结构化方法的指导思想是自顶向下、逐步求精的。其基本原则是功能的分解与抽象，它是软

件工程中最早出现的开发方法，特别适合于数据处理领域的问题。结构化方法对于规模大的项目，特别是复杂的项目不太适应，该方法难于解决软件重用问题，且难于适应需求变化的问题和彻底解决维护问题。

2. Jackson 方法

Jackson 方法是一种面向数据结构的开发方法。Jackson 方法首先描述问题的输入、输出数据结构、分析其对应性，然后推出相应的程序结构，从而给出问题的软件过程描述。

Jackson 方法根据一个问题的数据结构与处理该问题数据结构的控制结构的相似性原理形成了最初的 JSP（Jackson Structured Programming）方法。

JSP 方法是以数据结构为驱动的，适合于小规模的项目。当输入数据结构与输出数据结构无对应关系时，难于应用此方法，基于 JSP 方法的局限性，又发展了 JSD（Jackson System Development）方法。JSD 方法是一个完整的系统开发方法。它是以事件为驱动的，是一种基于进程的开发方法。它适合应用于时序特点较强的系统，包括数据处理系统和一些实时控制系统。

利用 JSD 方法开发系统时的步骤如下：

01 建立现实世界的模型。
02 确定系统的功能需求。
03 对需求的描述特别强调操作之间的时序性。

JSD 方法的缺点如下：

- 对客观世界及其同软件之间的关系认识不完整。
- 所确立的软件系统实现结构过于复杂。
- 软件结构说明的描述采用第三代语言，不利于软件开发者对系统的理解以及开发者之间的通信交流。

3. 维也纳开发方法（VDM）

维也纳开发方法（VDM）是一个基于模型的方法。

- 它的主要思想是将软件系统当作模型来给予描述，把软件的输入、输出看作是模型对象，把这些对象在计算机内的状态看作是该模型在对象上的操作。
- 它的目的是从软件系统最高一级抽象，直到最后生成目标的每一步都给予形式说明，以此提高软件的可靠性。

4. 面向对象的开发方法

面向对象的开发方法是以对象作为最基本的元素，它是分析问题、解决问题的核心。计算机实现的对象与现实世界的对象有一一对应的关系，不必做任何转换，这就使面向对象易于被人们所理解、接受和掌握。

面向对象的开发方法主要有 Booch 方法、Coad 方法和 OMT 方法等。为了统一各种面向对象方法的术语、概念和模型，1997 年推出了统一建模语言，即 UML（Unified Modeling Language）语言。它是面向对象的标准建模语言，通过统一的语义和符号表示，使各种方法的建模过程和表示

统一起来，将成为面向对象建模的工业标准。下面主要从面向对象方法的概念、面向对象方法的特点、面向对象方法的问题三个方面进行介绍。

（1）面向对象方法的概念

面向对象方法（Object-Oriented Method，OOM）是由面向对象程序设计（Object-Oriented Programming，OOP）发展起来的。面向对象方法基于类和对象的概念，把客观世界的一切事物都看成是由各种不同的对象组成，每个对象都有各自内部的状态、机制和规律，按照对象的不同特性，可以组成不同的类。不同的对象和类之间的相互联系和相互作用就构成了客观世界中的不同的事物和系统。

面向对象方法与生命周期法、原型法有很大的不同，生命周期法与原型法虽然在系统开发的阶段划分和顺序上各有特点，但基本上可以被称为是面向数据或面向过程的，即在获得基本的系统需求后要从该需求提炼出数据流或把需求转换为过程，而不是直接在客观需求上展开工作，系统需求与系统分析、设计与实现是不一致的，由此引发了一系列问题和隐患；而面向对象方法直接从系统需求出发，把需求分解成对象和类，数据和操作都“隐藏”于对象之中，通过对对象的定义、操纵来实现系统，从而达到系统需求与系统分析、设计与实现的一致。

面向对象方法的系统开发是在对系统调查和需求分析的基础上进行的，其工作过程可分为以下三个阶段。

- 面向对象分析阶段（Object-Oriented Analysis，OOA）：在当前要求解的复杂问题中，抽象地识别出对象及其行为、结构、属性方法等，建立未来信息系统的分析模型。
- 面向对象设计阶段（Object-Oriented Design，OOD）：对 OOA 阶段得到的分析模型进一步抽象、归类、整理，并规范化地形成基本的“类”，再模拟应用对象的特性，将对象的“事件”和“方法”封装进入这些“类”中，形成具有生命活力的智能“组件”。
- 系统实现阶段：用面向对象的程序设计语言将 OOD 阶段过程的范式直接映射为应用程序软件。

（2）面向对象方法的特点

面向对象方法具有以下特点：

- 系统开发的基础统一于对象之上，各个阶段的工作过渡平滑，避免了许多中间转换环节和多余劳动，加快了系统开发的进程，提高了系统开发的正确性和效率。
- OOA 方法的分析与结构化分析有较大的区别，前者强调的是在系统调查资料的基础上对所需素材进行归类分析和整理，而后者则是对管理业务现状和方法的分析。
- OOD 方法是根据对象来组织信息系统的逻辑结构，与 OOA 不同，OOD 阶段必须考虑系统实现，OOD 与系统是两个相互交织在一起的过程。
- 面向对象技术中的各种概念和特性，如继承、封装、多态性及消息传递机制等，使软件的一致性、模块的独立性以及程序的共享和可重用性大大提高。

（3）面向对象方法的问题

虽然面向对象方法在可重用性、系统可维护性和可理解性方面有着突出的优势，但也存在不足或局限，主要表现在以下几个方面。

- 容易造成系统结构不合理、各部分关系失调：与原型法类似，在大型的管理信息系统开发过程中如果不经过自顶向下的整体划分，而是一开始就自底向上的采用面向对象方法开发系统，同样也会造成系统结构不合理、各部分关系失调等问题。
- 用户直接参与较为困难：因为每个使用者一般只熟悉自己的日常工作的事物流程，对于系统中的对象没有总体的把握，因此面向对象方法也要求参与用户最好是问题域专家，而不仅仅是使用者。
- 需要一定的软件基础支持：系统实现的工具要么是某种程序设计语言，要么是某种面向对象的数据库程序设计语言，或是其他的一些繁杂的程序设计工具，无论是使用什么工具，系统实现总是一个程序设计的过程，因此，对于当今人们的即时优化的系统要求来说，面向对象的系统设计方法仍然显得束手无策、苍白无力。

近年来，系统软件日趋标准、完善，相比之下，应用软件生产方式在开发速度、可用性、兼容性、可扩充性、可重用性等重要方面都不能令人满意。而应用软件无论采用以上介绍的任一种方法开发都离不开编程，从而使得应用软件开发成本飞涨，软件质量跟不上时代发展的需要，供不应求的趋势十分强烈，这就是所谓的应用软件危机。时代要求寻找更高效、产品质量更高、更易维护且能即时优化的应用软件开发方法。

1.1.5 软件生存周期过程

软件生存周期概念的出现可以帮助我们较为全面地认识软件开发。在 1998 年制订和公布的国家标准《GB 8566-88 计算机软件开发规范》中，将软件生存周期划分为 8 个阶段，即可行性研究和计划、需求分析、概要设计、详细设计、实现、组装测试、确认测试、使用和维护。该标准为每个阶段规定了任务、实施步骤、实施要求以及完成的标志。对软件生存期按此方式进行 8 个阶段的划分大致符合也适应瀑布模型的要求。

20 世纪 90 年代初提出了软件工程过程的概念。软件工程过程规定了获取、供应、开发、操作和维护软件时，要实施的过程、活动和任务。其目的是为各种人员提供一个公共的框架，以便用相同的语言进行交流。该框架由几个重要的过程组成，这些主要过程含有用来获取、供应、开发、操作和维护软件所用的基本的、一致的要求。该框架还用于控制和管理软件过程。各种组织和开发机构都可以根据具体情况进行选择和剪裁。可在一个机构的内部或外部实施。

1995 年国际标准化组织在此基础上对生存期过程进行了调整，公布了新的国际标准，即《ISO/IEC 12207 信息技术——软件生存期过程》。该标准全面、系统地阐述了软件生存期的过程、活动和任务。标准定义的 17 个过程分别属于主要过程、支持过程和组织过程。可以通过图 1-5 了解它的结构。

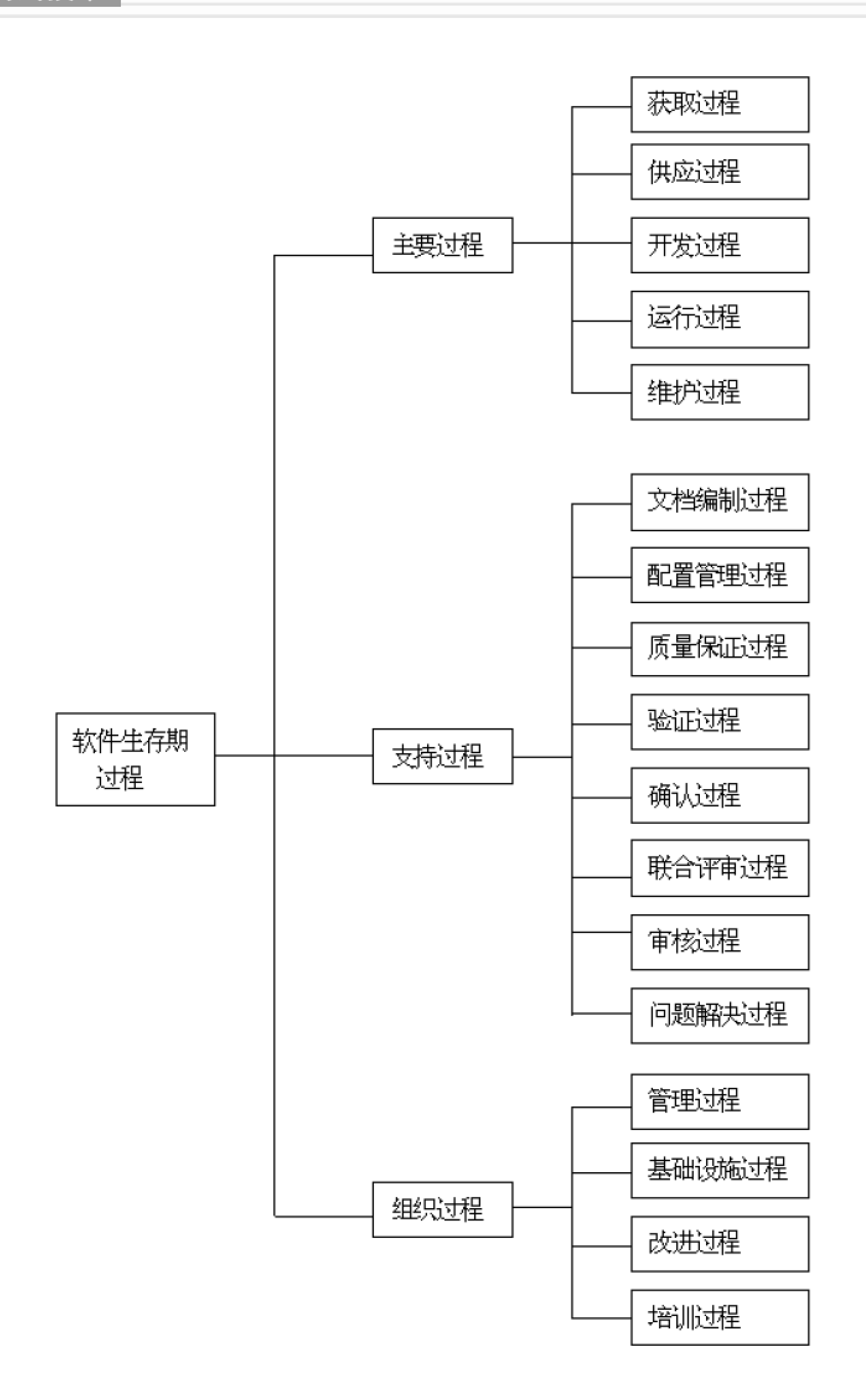


图 1-5　ISO/IEC 12207 信息技术——软件生存期过程

软件生存期过程的主要活动和任务描述如表 1-2 所示。

表 1-2 软件生存期过程的主要活动和任务描述

过程名		主体	主要活动和任务描述
主要过程	获取 Acquisition	需方	定义、分析需求或委托供方进行需求分析而后认可；招标准备；合同准备以及验收
	供应 Supply	供方	评审需求；准备投标；签订合同；制订并实施项目计划；开展评审及评价；交付产品
	开发 Development	开发者	系统需求分析；系统结构设计；软件需求分析；软件结构设计；软件详细设计；软件编码和测试；软件集成、软件合格测试；系统集成；系统合格测试、软件安装及软件验收支持
	运行 Operation	运行者	制定并实施运行计划；运行测试；系统运行；对用户提供帮助和咨询
	维护 Maintenance	维护者	问题和变更分析；实施变更；维护评审及维护验收；软件移植及软件退役
支持过程	文档编制 Documentation		设计文档编制标准；确认文档输入数据的来源和适宜性；文档的评审及编辑；文档发布前的批准；文档的生产与提交、储存和控制；文档的维护
	配置管理 Configuration Management		配置标识；配置控制；记录配置状态；评价配置；发行管理与交付
	质量保证 Quality Assurance		软件产品的质量保证；软件过程的质量保证以及按 ISO 9001 标准实施的质量体系保证
	验证 Verification		合同、过程、需求、设计、编码、集成和文档等的验证
	确认 Validation		为分析测试结果实施特定的测试；确认软件产品的用途；测试软件产品的适用性
	联合评审 Joint Review		实施项目管理评审（项目计划、进度、标准、指南等的评价）；技术评审（评价软件产品的完整性、符合标准等）
	审核 Audit		检验项目是否符合需求、计划、合同以及规格说明和标准
	问题解决 Problem Resolution		检验和解决开发、运行、维护或其他过程中出现的问题，提出响应对策，使问题得到解决
组织过程	管理 Management	管理者	制定计划、监控计划的实施、评价计划实施，涉及到有关过程的产品管理、项目管理和任务管理
	基础设施 Infrastructure		为其他过程所需的硬件、软件、工具、技术、标准，以及开发、运行或维护所用的各种基础设施提供建立和维护服务
	改进 Improvement		对整个软件生存期过程进行评估度量、控制和改进
	培训		制订培训计划；编写培训资料；培训计划的实施

在表 1-2 中给出了 17 个过程的主要活动和任务的描述。以下对该标准提出的软件生存期过程给予简要说明。

1. 主要过程（Primary Process）

主要过程包括 5 个过程，这些过程供各主要当事方（如需方、供方、开发者、运行者和维护者）在参与或完成软件产品开发、运行或维护时使用，它们是获取过程、供应过程、开发过程、运行过程、维护过程。

- 获取过程：需方获取系统、软件产品或软件服务的活动。
- 供应过程：供方向需方提供系统、软件产品或软件服务的活动。
- 开发过程：开发者定义并开发软件产品的活动。
- 运行过程：运行者在规定的环境中为其用户提供计算机系统服务的活动。
- 维护过程：维护者提供维护软件产品服务的活动。

2. 支持过程（Supporting Process）

支持过程包括 8 个过程，其每个过程均有明确的目的支持其他过程，帮助软件项目获得成功及良好的产品质量。它们是文档编制过程、配置管理过程、质量保证过程、验证过程、确认过程、联合评审过程、审核过程、问题解决过程。

- 文档编制过程：记录生存期过程中产生的信息所需要的活动。
- 配置管理过程：实施配置管理活动。
- 质量保证过程：为确保软件产品和软件过程符合规定的需求并能坚持既定计划所需要的活动。联合评审、审核、验证与确认可作为质量保证技术使用。
- 验证过程：为确保一个活动的产品满足前一活动对它的要求和条件的活动。
- 确认过程：为确保最终产品满足预期使用要求的活动。
- 联合评审过程：评审方与被评审方共同对某一活动的状态和产品进行评审的活动。
- 审核过程：审核项目是否按要求、计划、合同完成的活动。
- 问题解决过程：分析和解决在开发、运行、维护或其他过程中出现的总是不论其性质和来源如何的活动。

3. 组织过程（Organizational Process）

组织过程包括 4 个过程，这些过程被某个机构用来建立和实现与生存期过程相关的基础结构，甚至人事制度，并使其不断改进，它们是管理过程、基础设施过程、改进过程、培训过程。

- 管理过程：规定生存期过程中的基本管理活动，包括项目管理。
- 基础设施过程：建立生存期过程基础结构的基本活动。
- 改进过程：某一机构（需用、供方、开发者、运行者、维护者或其他过程的管理者）为建立、测量、控制和改进其生存期过程所需要开展的基本活动。
- 培训过程：对人员进行适当培训所需要的活动。

1.1.6 软件工程标准

根据软件工程标准制定的机构和标准适用的范围有所不同，它可分为国际标准、国家标准、行业标准。

1. 国际标准

国际标准由国际联合机构制定和公布，提供各国参考的标准，如 ISO（International Standards Organization）——国际标准化组织。这一国际机构有着广泛的代表性和权威性，它所公布的标准也有较大影响。在 1960 年初，该机构建立了“计算机与信息处理技术委员会”，简称 ISO/TC97，专门负责与计算机有关的标准化工作。这一标准通常冠有 ISO 字样，如 ISO 8631-86 Information Processing-Program Constructs and Conventions for Their Representation《信息处理——程序构造及其表示法的约定》。该标准现已由我国收入国家标准。

2. 国家标准

国家标准是指由国家制定或批准的标准，适用于全国范围的标准。我国已陆续制定和发布了 20 项国家标准。这些标准可分为 4 类：

- 基础标准。
- 开发标准。
- 文档标准。
- 管理标准。

对 4 类标准的详细说明如表 1-3 所示。

表 1-3 对 4 类标准的详细说明

分类	标准名称	标准号	
基础标准	信息处理——数据流程图、程序流程图、系统流程图、程序网络图和系统资源图的文件编辑符号及约定	GB 1526－89	ISO 5807－1985
	软件工程术语	GB/T 11457－89	
	软件工程标准分类法	GB/T 15538－95	ANSI/IEEE 1002
	信息处理——程序构造及其表示法的约定	GB 13502－92	ISO 8631
	信息处理——判定表的规范	GB/T 15535－95	ISO 5806
	信息处理系统——计算机系统配置图符号及其约定	GB/T 14085－93	ISO 8790
开发标准	软件开发规范	GB 8566－88	
	计算机软件单元测试	GB/T 15532－95	
	软件支持环境		
	信息处理——按记录组处理顺序文卷的程序流程		ISO 6593－1985
	软件维护指南	GB/T 14079－93	
文档标准	软件文档管理指南		
	计算机软件产品开发文件编制指南	GB 8567－88	
	计算机软件需求说明编制指南	GB 9385－88	ANSI/IEEE 829
	计算机软件测试文件编制规范	GB 9386－88	ANSI/IEEE 830
管理标准	计算机软件配置管理计划规范	GB/T 12505－90	IEEE 828
	信息技术、软件产品评价、质量特性及其使用指南	GB/T 12260－96	ISO/IBC 9126－91
	计算机软件质量保证计划规范	GB 12504－90	ANSI/IEEE 730
	计算机软件可靠性和可维护性管理	GB/T 14394－93	
	质量管理和质量保证标准的第三部分：GB/T 19001－ISO 9001 在软件开发、供应和维护中的使用指南	GB/T 19000.3－94	ISO 9000－3－93

3. 行业标准

行业标准是指由行业机构、学术团体或国防机构制定，并适用于某个业务领域的标准，如 IEEE、GJB 等。

- IEEE（Institute of Electrical and Electronics Engineers）：美国电气与电子工程师学会。近年来该学会专门成立了软件标准分技术委员会（SESS），积极开展了软件标准化活动，取得了显著成果，受到了软件界的关注。IEEE 通过的标准经常要报请 ANSI 审批，使之具有国家标准的性质，因此，日常看到 IEEE 公布的标准常冠有 ANSI 的字头，例如 ANSI/IEEE Str 828-1983《软件配置管理计划标准》。
- GJB：中华人民共和国国家军用标准。这是由中国国防科学技术工业委员会批准，适合于国防部门和军队使用的标准，例如，1988 年实施的 GJB 437-88《军用软件开发规范》、GJB 438-88《军用软件文档编制规范》。

1.1.7 软件开发文档

在软件的开发过程中，一般来说，应该产生如下 14 种文件：

- 可行性研究报告。
- 项目开发计划。
- 软件需求说明书。
- 数据要求说明书。
- 概要设计说明书。
- 详细设计说明书。
- 数据库设计说明书。
- 用户手册。
- 操作手册。
- 模块开发卷宗。
- 测试计划。
- 测试分析报告。
- 开发进度月报。
- 项目开发总结报告。

在软件开发过程中，承建单位需要根据软件关键等级和软件规模等级的不同选择文档。

1.1.8 软件质量保证

软件的质量是长期以来困扰软件业发展的一个大问题。生产出高质量的软件产品是软件工程等学科研究的主要目标。

在计算机发展的早期，软件质量保证曾经只由程序员承担。在今天，这一定义的含义是在一个组织中有多个机构负有保证软件质量的责任，包括软件工程师、项目管理者、客户、销售人员。软件质量保证由一种应用于整个软件过程的保护性活动。它的内容很多，主要包括以下几项：

- 质量管理方法。
- 有效的软件工程技术（方法和工具）。
- 在整个软件过程中采用的正式技术评审。
- 多层次的测试策略。
- 对软件文档及其修改的控制。
- 保证软件遵守软件开发标准的规程（在适用时）。
- 度量和报告机制。

为了从事软件质量保证活动，需要在软件企业建立起质量保证体系，并往往需要第三方的认证。目前的 ISO9000 族的标准和美国卡耐基梅隆大学软件工程研究所的过程能力成熟度模型等为软件企业建立质量体系和进行软件质量保证工作提供了指导。

1. 软件质量

在进行软件质量保证时，首先需要对软件质量有正确的完整的理解，并且通过定性或定量的方法对软件质量进行度量。

一个对软件质量进行系统评估的框架分为以下三层：

- 从用户的角度衡量软件质量。
- 从开发者的角度看待软件质量。
- 更细化的、可定量衡量的指标。

软件质量可分为正确性、可靠性、功效、完整性、可用性、可维护性、灵活性、可测试性、可移植性、可复用性、互操作性。

- 正确性：一个程序满足它的需求规格和实现用户任务目标的程度。
- 可靠性：一个程序期望以所需要的精确度完成它的预期功能的程度。
- 功效：一个程序完成其功能所需要的计算资源和代码的数量。
- 完整性：对未授权人员访问软件或数据的可控制程度。
- 可用性：学习、操作、准备输入和解释程序输出所需要的工作量。
- 可维护性：定位和修复程序中一个错误所需要的工作量。
- 灵活性：修改一个运作的程序所需要的工作量。
- 可测试性：测试一个程序以确保它完成所期望的功能所需要的工作量。
- 可移植性：把一个程序从一个硬件或软件系统环境移植到另一个环境所需要的工作量。
- 可复用性：一个程序（或一个程序的一部分）可以在另外一个应用程序中复用的程度。
- 互操作性：连接一个系统和另一个系统所需要的工作量。

2. 软件过程能力成熟度模型（CMM）

软件产品的质量取决于软件开发过程，具有良好软件过程的软件机构能够开发出高质量的软件产品。1987 年在美国国防部的支持下，卡耐基梅隆大学推出了软件过程评估项目的研究成果——软件过程能力成熟度模型（Capacity Maturity Mode1，CMM）。该模型在软件界引起了广泛的关注，以至在其基础上形成了国际标准（ISO/IEC 15504）。

CMM 提供了一个框架，将软件过程改进的进化步骤组织成 5 个成熟度等级，如表 1-4 所示，

为过程不断改进奠定了循序渐进的基础。每一个成熟度等级为持续改进过程提供一个台阶。每一等级包含一组过程目标，通过实施相应的一组关键过程域达到这一组过程目标，当目标满足时，能使软件过程的一个重要成分稳定。

表 1-4　CMM 提供的 5 个成熟度等级

过程能力等级	特点	关键过程域
初始级	软件过程是无序的，有时甚至是混乱的，对过程几乎没有定义，成功取决于个人努力。管理是反应式（消防式）的	
重复级	建立了基本的项目管理过程来跟踪费用、进度和功能特性。制订了必要的过程纪律，能重复早先类似应用项目取得的成功	需求管理、软件项目策划、软件项目跟踪和监督、软件子合同管理、软件质量保证、软件配置管理
已定义级	已将软件管理和工程两个方面的过程文档化、标准化，并综合成该组织的标准软件过程。所有项目均使用经批准、剪裁的标准软件过程来开发和维护软件	组织过程定义、组织过程焦点、培训大纲、集成软件管理、软件产品工程、组织协调、同行专家评审
已定量管理级	收集对软件过程和产品质量的详细度量，对软件过程和产品都有定量的理解与控制	定量的过程管理、软件质量管理
优化级	过程的量化反馈和先进的新思想、新技术促进过程不断改进	缺陷预防、技术变更管理、过程变更管理

CMM 的级别示意图如图 1-6 所示。

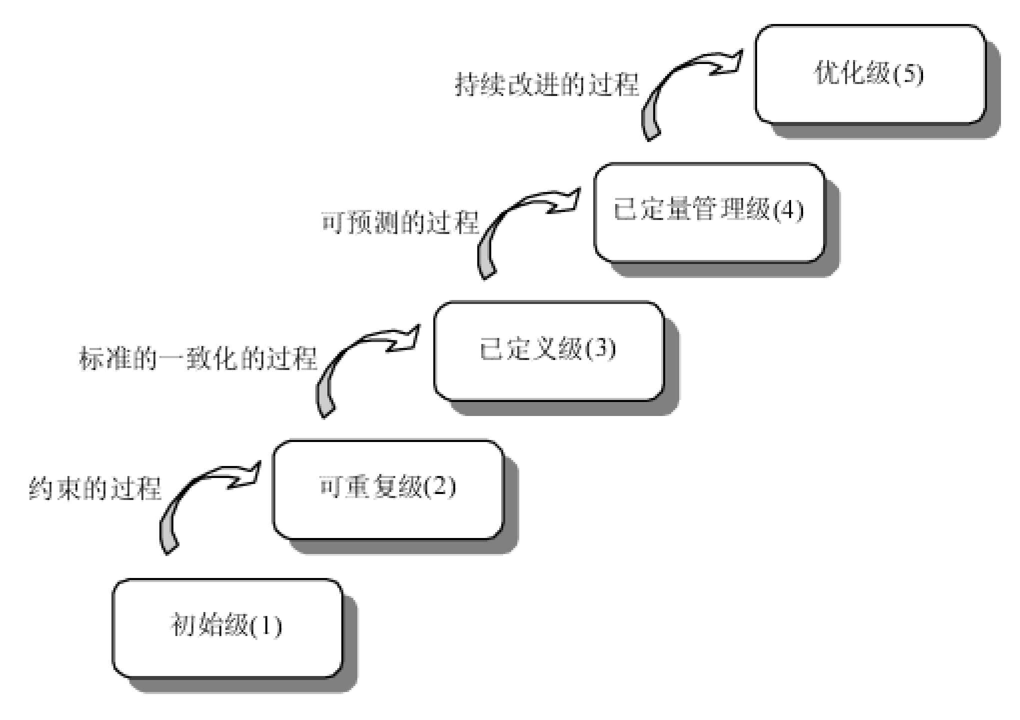


图 1-6　CMM 的级别示意图

由于 CMM 是专门针对软件企业的，因此其针对性好，在实施的时候不需要进行术语上的转换。另外，其逐步提高的方式使得其可操作性更强。

3. 实现能力成熟度模型集成（CMMI）

卡内基梅隆大学在 2001 年 9 月又推出了比较成熟的系统工程和软件工程的实现能力成熟度模型集成CMMI（Capability Maturity Model Integration）。这个模型可以指引一个组织去改进它用于开发、维护、购买产品和服务的过程。该模型包括了连续模型和阶段模型这两种表示方法，示意图如图 1-7 和 1-8 所示。

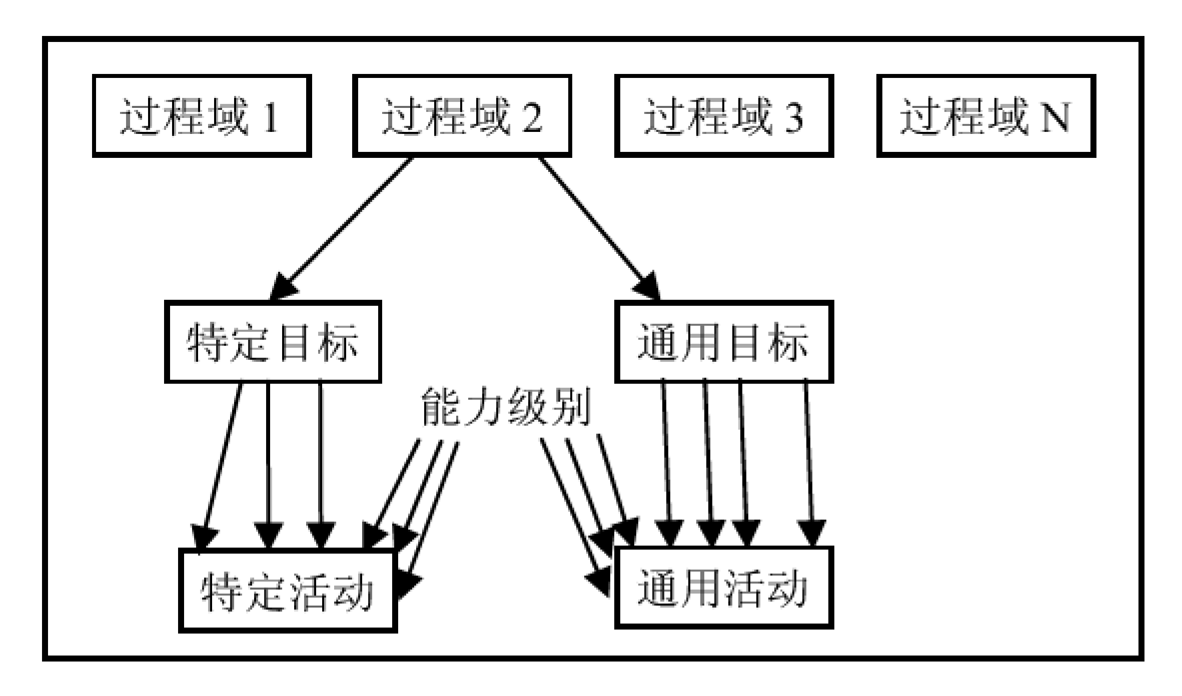


图 1-7　连续模型示意图

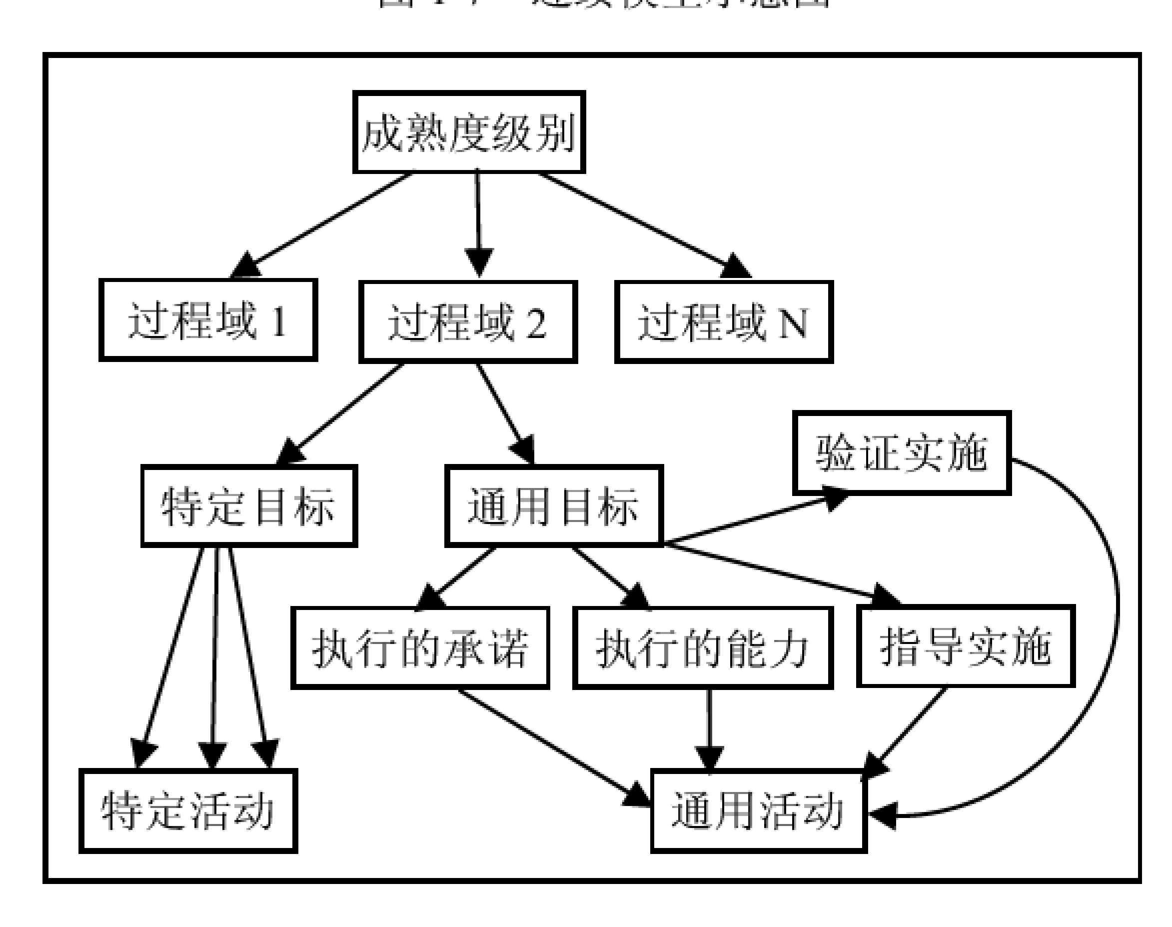


图 1-8　阶段模型示意图

一个组织可以根据自己的过程改进要求来自由选择合适的表示方法并使用。

在 CMMI 模型中，这两种表现方式（连续的和阶段的）从它们所涵盖的过程区域上来说并没有什么不同，不同的是过程区域的组织方式以及对成熟度（能力）级别的判断方式。熟悉 CMM 模型的人会发现，在 CMMI 模型中将目标和名称进行了改变。

- 名称上的改变：将关键过程区域（KPA）转变成了过程区域（PA）。
- 将每一个 PA 的目标分成了通用目标（Generic Goals）和特定目标（Specific Goals）。这是对 CMM 中关于制度化部分的一种强调，而且正是因为有了这个变化，在连续模型中出现了 0 级和 1 级的区分。针对目标的细分，实践活动也进行了细分。活动也变成了通用活动和特定活动两种。

对于连续模型中每一个过程区域都具有 6 个级别（0~5），其说明如表 1-5 所示。

表 1-5　连续模型中每一个过程区域的 6 个级别

级别	Name	名称	特性
0	Incomplete	不完整	过程未执行或者执行不完整，特定目标中有不能满足的部分
1	Performed	已执行	特定目标都得到满足，基本活动都得到执行

（续表）

级别	Name	名称	特性
2	Managed	已管理	已管理的过程除了得到执行外，还需要得到计划，并且按照组织方针来进行实施，相关人员得到与执行有关的培训，为了过程的执行分配了相关的资源，生成的工作产品受到控制。利益相关的方面都参与了过程的执行，并且进行了相关的评审以及过程符合度的验证。管理层关心过程的制度化状况以及过程的其他目标，例如成本、日程和质量目标
3	Defined	已定义	已定义的过程除了是一个已管理的过程之外，还具有如下的特征：该过程是从组织的标准过程裁减而来的，裁减的依据是组织的裁减指南。该过程还向组织的过程资产库贡献关于工作产品、度量数据以及过程的其他目标，例如成本、例程和质量目标
4	Quantitatively Managed	量化管理	量化管理的过程除了是已定义的过程之外，还具有如下的特征：过程是使用统计的以及其他种类的量化手段来进行管理的；在过程的管理中使用了量化的质量和过程性能指标作为管理的标准；用统计手段来理解质量和过程性能，并且在整个生命周期之内进行管理
5	Optimizing	优化	优化的过程除了是一个量化管理的过程之外，还具有如下的特征：过程能够得到及时的变量和采用来满足当前的或者预期的业务目标；优化的过程聚焦于使用增量和创新技术进步手段来达到不断改进过程性能的目的；过程性能偏差的根本原因得到识别、并且针对这些原因采取相应的改进措施。这些措施按照一种能够度量的方式被识别、评价和实施。这些改进措施的选择是基于对组织过程的量化理解，以及这些改进措施的预期收益、成本以及影响程度。优化过程的性能能够不断地提高

连续模型中可以为每一个过程区域判定能力级别，这样就可以避免在 CMM 模型中那样，就算其他的领域都做得很好，只是因为有一个 KPA 中有一个目标不能达到就使得评估结果只能是 1 级的尴尬局面。连续模型和阶段模型中的过程元素都是一致的，在连续模型中，过程区域是按照过程区域分类来划分的。

在连续模型中，主要包括下面的过程区域分类和过程区域，如表 1-6 所示。

表 1-6　在连续模型中的过程区域分类和过程区域

过程区域类型	过程区域	简要描述
过程管理	组织过程焦点	组织根据对自身过程优劣之处的了解，计划和实施组织级的过程改进活动
	组织过程定义	组织建立和维护一个可用的组织过程资产库
	组织培训	开发组织中人员的技能和知识，使相关人员获得有效履行职责的能力
	组织过程性能	建立和维护对组织标准过程性能的量化理解，理解组织标准过程对质量以及过程性能目标的支持，提供过程性能数据、基线以及项目量化管理的模型
	组织创新和部署	选择和部署对组织的过程和技术进行增量和创新性的改进的可以度量的方法。改进活动支持组织的质量和过程性能目标
	集成项目管理	项目通过对标准流程进行裁减开发项目的管理过程，按照这个流程与相关个人和组织对项目进行管理
	风险管理	对项目的潜在问题进行识别，以便在项目整个生命周期内对处理这些问题制定计划，以避免或者减少潜在问题的影响或者发生概率
	量化项目管理	对项目的过程进行量化的管理，以便达成项目所制定的质量和过程性能目标

（续表）

过程区域类型	过程区域	简要描述
工程	需求管理	管理项目中产品以及产品构件的需求，识别项目需求和计划、产品之间的差异
	需求开发	开发和分析客户需求，产品需求和产品构件的需求
	技术方案	设计、开发和实施对需求的解决方案。解决方案、设计和实施包括了产品、产品组件，以及与产品相关的生命周期过程中的一个或者它们的组合
	产品集成	将产品组件组装成产品，确保集成后的产品正常工作，并且发布该产品
	验证	确保开发出来的产品满足需求规格说明的要求
	确认	证明产品及其组件在预想的环境中能够按照预想的功能工作
支持	配置管理	创建和维护产品的完整性，包括配置识别、配置控制、配置状态统计、配置审计活动
	过程和产品质量保证	向管理层和员工提供产品和过程质量的可见度
	度量和分析	开发和维护组织的度量能力，给组织提供所需的支持管理信息
	决策分析和决议	使用正式的评价过程对识别出的可能的决策进行分析，按照制订的标准选择合适的决策
	原因分析和决议	识别缺陷和其他问题的原因，采取行动避免这些问题的缺陷和发生

与 CMM 模型相同，在 CMMI 的阶段模型中也划分了 5 个成熟度级别，对 5 个成熟度级别所包含的过程区域的详细说明如表 1-7 所示。

表 1-7　CMMI 阶段模型中的 5 个成熟度级别

级别	名称	包含过程区域
1	初始	无
2	已管理	需求管理 项目计划 项目监督和控制 供应商协议管理 度量和分析 过程和产品质量保证 配置管理
3	已定义	需求开发 技术解决方案 产品集成 验证 确认 组织过程焦点 组织过程定义 组织培训 集成项目管理 风险管理 决策分析和决议

（续表）

级别	名称	包含过程区域
4	量化管理	组织过程性能 量化项目管理
5	优化	组织创新和部署 原因分析和决议

在这里可以发现，在 CMMI 中，对系统工程进行了特别的强调，在 3 级中的需求开发、技术解决方案、产品集成、验证都是与系统工程直接密切相关的过程区域。

CMMI 模型的前身是 SW-CMM 和 SE-CMM，前者是关于软件开发的能力成熟度模型，后者是系统工程的模型。在这里，为了加深理解，把 CMMI 与 SW-CMM 进行一个初步的比较（以下提到的 CMM 均为 SW-CMM）。

- 在 CMMI 模型中出现了连续模型。这就允许参与评估的组织能够用一种更加灵活的方式来对自己的过程进行评估。在 CMMI 的评估中，可以选择需要的 PA（过程区域——Process Area），并针对每一个 PA 分别评判级别。这样可以帮助一个组织以及这个组织的客户更加客观和全面地了解它的过程成熟度，而不是像 CMM 一样一个数字决定一切。同时，连续模型的采用可以给一个组织在进行过程改进的时候带来更大的自主性，不用再像 CMM 一样受到级别的严格限制。这种改进的好处是灵活性和客观性强，弱点在于由于缺乏指导，一个组织可能缺乏对 PA 之间依赖关系的正确理解而片面的实施过程，造成一些过程成为空中楼阁，缺少其他过程的支撑。
- CMMI 模型中比 CMM 进一步强化了对需求的重视，也就是说，强调对有质量的需求进行管理，而如何获取需求则没有提出明确的要求。在 CMMI 的阶段模型中，3 级有一个独立的 PA 叫做需求开发，提出了对如何获取优秀的需求的要求。
- CMMI 模型对工程活动进行了一定的强化。在 CMM 中，只有 3 级中的软件产品工程和同行评审两个 KPA 是与工程过程密切相关的，而在 CMMI 中，则是将需求开发、验证、确认、技术解决方案、产品集成作为单独的 PA 进行了要求，从而在实践上提出了对工程的更高要求和更具体的指导。CMMI 中还强调了风险管理。不像在 SW-CMM 中把风险的管理分散在项目计划和项目跟踪与监控中进行要求，CMMI 的 3 级里包含了一个独立的 PA，叫做风险管理。
- 因为 CMM 中对度量与分析的要求比较笼统和空泛，CMMI 中将度量和分析作为一个独立的 2 级 PA 进行要求，从而强调了量化管理这个方面。

1.2 软件测试的概念、方法和任务

1.2.1 软件测试的概念

软件测试是软件工程中的一个环节，是开发项目整体的一部分。软件测试是有计划、有组织的，是保证软件质量的一种手段，它是软件工程中一个非常重要的环节，因此可以认为它是伴随软件工程的诞生而诞生的，伴随着软件复杂程度的增加、规模的增大，软件测试作为一种能够保证软

件质量的有效手段，越来越受到人们的重视，软件测试最终的目的是使产品达到完美。

软件测试方法没有完全标准化和统一化，因为从软件产业的产品到软件测试都有各式各样的软件，这里介绍的软件测试方法可用于多数应用软件的测试。

软件测试不是万能的，不可能发现全部的软件缺陷，而且软件的功能和性能不是由测试决定的，软件测试是有局限性的。

软件测试是在完成程序设计阶段工作后，经程序编码员测试已初步奠定了基础，进一步完成的测试工作。软件要投入运行前是否正确无误这一点极其重要，所以软件要在投入运行前施行测试。未经周密测试的软件贸然投入运行，将会造成难以想象的后果。

尽管各个程序在设计完毕以后都一一作过调试，但能否发挥整个系统的功能尚不清楚，例如某一程序运行与后续的程序运行是否矛盾？能否顺利连续？整个系统的总测试要等全部程序设计结束且能连续测试时才可进行。

由于软件错误的复杂性，长期以来，人们对软件测试的认识一直是模糊的。许多科学家从不同的角度给出了软件测试的不同定义，但总体来看都是不全面的。给软件带来错误的原因有很多，具体地说主要有如下几点：

- 交流不够、交流上有误解或者根本不进行交流。
- 应用需求不清晰的情况下进行开发。
- 软件复杂性（图形用户界面、客户/服务器结构、分布式应用、数据通信、关系型数据库使得软件及系统的复杂性呈指数增长）。
- 程序设计错误。
- 需求变化。
- 代码文档贫乏等。

软件测试可被认为：

- 软件测试是为了发现错误而执行程序的过程。
- 使用人工或自动手段来运行或测定某个系统的过程，其目的在于检验它是否满足规定的需求或是弄清楚预期结果与实际结果之间的差别。
- 测试是为了证明程序有错，而不是证明程序无错误。
- 一个好的测试用例是在于它能发现至今未发现的错误。
- 一个成功的测试是发现了至今未发现的错误的测试。
- 测试是以查找错误为中心，而不是演示软件的正确功能。
- 测试并不仅仅是为了要找出错误，而是通过错误分析错误产生的原因和错误的分布特征，可以帮助项目管理者发现当前所采用的软件过程的缺陷，以便改进。

下面重点讨论软件测试的目的、软件测试的基本原则、软件测试的步骤、软件错误的分类、软件测试的工具。

1. 软件测试的目的

软件测试的不同机构会有不同的测试目的，相同的机构也可能有不同测试目的。

软件测试的目的决定了如何去组织测试。如果测试的目的是为了尽可能多地找出错误，那么

测试就应该直接针对软件比较复杂的部分或是以前出错比较多的位置。如果测试的目的是为了给最终用户提供具有一定可信度的质量评价，那么测试就应该直接针对在实际应用中会经常用到的商业假设。

测试可视为分析、设计和编码 3 个阶段的“最终复审”，在软件质量保证中具有重要地位。为了确保软件的质量，较理想的做法应该是对软件的开发过程，按软件工程各阶段形成的结果，分别进行严格的审查。

软件测试的目的具有如下几点：

- 帮助开发人员、测试工程师发现问题、分析问题。
- 减少软件的缺陷数目或者降低软件的缺陷密度。
- 提高软件的可靠性。
- 评估软件的性能指标。
- 增加用户对软件的信心。
- 测试的最终目的是为了避免错误的发生，确保应用程序能够正常高效的运行。

2. 软件测试的基本原则

为了提高测试的质量，软件测试需要遵循下面的原则。

- 制定严格的测试计划，并把测试时间安排得尽量宽松，不要希望在极短时间内完成一个高水平的测试。测试计划应包括：所测试软件的功能、输入和输出、测试内容、各项测试的进度安排、资源要求、测试资料、测试工具、测试用例的选择等。
- 应尽可能早地开始测试。在软件生命周期中，1 个错误发现的越晚，修复错误的费用越高。其说明如图 1-9 所示。

阶段	相对修复费用
需求阶段	0.1~0.2
设计阶段	0.5
编码阶段	1
单元测试阶段	2
验收阶段	5
维护阶段	20

图 1-9　修复错误的费用

- 测试应从“小规模”开始，逐步转向“大规模”。最初的测试通常把焦点放在单个程序模块上，进一步测试的焦点则转向在集成的模块簇中寻找错误，最后在整个系统中寻找错误。
- 测试用例应由测试输入数据、测试执行步骤和与之对应的预期输出结果三部分组成。
- 按照组件和功能特征的优先级从高到低的顺序进行测试。
- 重点放在处理多语言字符串的直接或间接的输入/输出（I/O）。
- 对测试错误结果一定要有一个确认的过程。
- 回归测试的关联性一定要引起充分的注意，修改一个错误而引起更多错误出现的现象并不少见。

- 妥善保存一切测试过程文档，意义是不言而喻的，测试的重现性往往要靠测试文档。妥善保存测试计划、测试用例、出错统计和最终分析报告，为维护提供方便。
- 在各种语言环境下安装应用程序。
- 通过各种区域设置卸载应用程序。
- 使用不同区域的输入法编辑器交互式文本输入。
- 进行多语言文本的剪贴板操作。
- 正确输入、存储并检索区域的特定数据。
- 验证带有数据分割符的输入时间、日期和数值。
- 程序员应该避免检查自己的程序，测试工作应该由独立的专业的测试软件和软件测试机构来完成。
- 测试用例的设计要确保能覆盖所有可能路径。
- 充分注意测试中的群集现象。经验表明，测试后程序残存的错误数目与该程序中已发现的错误数目或检错率成正比。应该对错误群集的程序段进行重点测试。
- 严格排除测试的随意性。
- 应当对每一个测试结果做全面的检查。
- 在设计测试用例时，应当包括合理的输入条件和不合理的输入条件。合理的输入条件是指能验证程序正确的输入条件，不合理的输入条件是指异常的、临界的、可能引起问题异变的输入条件。软件系统处理非法命令的能力必须在测试时受到检验。用不合理的输入条件测试程序时，往往比用合理的输入条件进行测试能发现更多的错误。

3. 软件测试的步骤

软件测试的步骤一般可按如表 1-8 所示的方式进行。

表 1-8　软件测试步骤

软件测试步骤	输入	输出
测试需求分析	1. 软件测试的方法与规范 2. 软件需求的规格说明 3. 软件设计说明（概要设计说明和详细设计说明）	软件测试计划： ● 软件测试的定位 ● 软件测试线索 ● 软件测试环境的定义 ● 软件需求的追踪矩阵
测试过程设计	1. 软件测试的方法与规范 2. 软件测试计划	软件测试说明： ● 软件测试步骤 ● 软件测试基准 ● 测试线索的追踪矩阵
测试实现	1. 软件测试的方法与规范 2. 软件测试说明 3. 软件测试工具	软件测试的实现配置： ● 软件测试环境 ● 测试步骤的计算机表示（用于回归测试的测试代码/测试数据） ● 测试基准的计算机表示

（续表）

软件测试步骤	输入	输出
测试实施	1. 软件测试的方法与规范 2. 软件测试说明 3. 软件测试的实现配置	软件测试记录： ● 测试运行结果的计算机表示 ● 测试比较结果的计算机表示 ● 测试日志 ● 软件问题报告
测试评价	1. 软件开发文档 2. 软件测试文档 3. 软件测试配置 4. 软件测试记录	软件测试报告： ● 测试结果的统计信息 ● 测试结果的分析/评判
测试配置管理	测试配置管理项： ● 软件测试的描述性表示（测试文档/文件） ● 软件测试的计算机表示（测试代码/数据/结果）	1. 软件测试配置管理项的标识管理 2. 软件测试配置管理项的存储管理 3. 软件测试配置管理项的引用控制 4. 软件测试配置管理项的版本控制 5. 软件测试配置管理项的改动控制
测试维护	测试配置管理项	1. 测试配置管理项的使用报告 2. 测试配置管理项的软件问题报告 3. 测试配置管理项的更动控制文件

明确区分测试步骤，并明确定义其资源（人/物/时间）的安排，是保障软件测试工作有序开展、有效管理的关键。

明确区分测试需求分析的步骤和测试过程设计的步骤的意义还在于：软件测试对软件功能/软件实现有了可追踪性，因而为准确评价测试用例的质量提供依据。

4. 软件错误的分类

软件错误存在于软件生存期的各个阶段，凡是和预期的状态或行为不相符合的统称为软件错误，例如制定的计划和实际不相符合的、需求说明书有问题的、设计说明书有问题的、程序编码有问题的、测试有问题的、运行结果不正确的、维护有问题的等都是软件错误。

软件的错误是非常复杂的，凡是人们能够想到的软件错误都是可能发生的。到目前为止，人们对软件错误的认识尚停留在表面上，其内在规律有待进一步研究。这也是目前软件测试技术发展比较缓慢的原因之一。

（1）软件的错误按性质分类

软件的错误按性质可分为：功能错误、系统错误、加工错误、数据错误、代码错误。

① 功能错误

功能错误主要包括以下几点：

- 规格说明不完全，有二义性或自身矛盾。
- 程序实现的功能与用户要求的不一致。
- 软件测试的设计与实施发生错误。
- 测试标准引起的错误。

② 系统错误

系统错误主要包括以下几点。

- 外部接口错误：外部接口是指如终端、打印机、通信线路等系统与外部环境通信的手段。所有外部接口之间、人与机器之间的通信都使用专门的协议。
- 内部接口错误：内部接口是指程序之间的联系。它所发生的错误与程序内实现的细节有关。
- 硬件结构错误。
- 操作系统错误。
- 软件结构错误：由于软件结构不合理或不清晰而引起的错误。
- 控制与顺序错误：控制与顺序错误忽视了时间因素而破坏了事件的顺序；猜测事件出现在指定的序列中；等待一个不可能发生的条件；漏掉先决条件；规定错误的优先级或程序状态；漏掉处理步骤；存在不正确的处理步骤或多余的处理步骤等。
- 资源管理错误：资源管理错误是由于不正确地使用资源而产生的，例如使用未经获准的资源；使用后未释放资源；资源死锁；把资源链接在错误的序列中等。

③ 加工错误

加工错误主要包括以下几点。

- 算术与操作错误：算术与操作错误是指在算术运算、函数求值和一般操作过程中发生的错误，包括数据类型转换错、除法溢出、错误地使用关系比较符、用整数与浮点数进行比较等。
- 初始化错误：初始化错误主要是指忘记初始化工作区；忘记初始化寄存器和资料区；错误地对循环控制变量赋初值；用不正确的格式、数据或类型进行初始化等。
- 控制和次序错误：控制和次序错误主要是指遗漏路径；不可达到的代码；不符合语法的循环嵌套；循环返回和终止的条件不正确；漏掉处理步骤或处理步骤有错等。
- 静态逻辑错误：静态逻辑错误主要是指不正确地使用 CASE 语句；在表达式中使用不正确的否定（例如用“>”代替“<”的否定）；对情况不适当地分解与组合；混淆“或”与“异或”等。

④ 数据错误

数据错误只要是指以下几点。

- 动态数据错误：动态数据是在程序执行过程中暂时存在的数据。各种不同类型的动态数据在程序执行期间将共享一个共同的存储区域，若程序启动时对这个区域未初始化，就会导致数据出错。
- 静态数据错误：静态数据在内容和格式上都是固定的。它们直接或间接地出现在程序或数据库中。由编译程序或其他专门程序对它们进行预处理。
- 数据内容错误：数据内容是指存储于存储单元或数据结构中的位串、字符串或数字被破坏或被错误地解释而造成的错误。
- 数据结构错误：数据结构是指数据元素的大小和组织形式。在同一存储区域中可以定义不同的数据结构。数据结构错误主要包括结构说明错误及把一个数据结构误当做另一类数据结构使用的错误。

- 数据属性错误：数据属性是指整数、字符串、子程序等。数据属性错误对数据属性不正确地解释，如错把整数当实数、允许不同类型的数据混合运算而导致的错误等。

⑤ 代码错误

代码错误主要包括：语法错误、打字错误、对语句或指令不正确理解所产生的错误等。

（2）软件的错误按生存期阶段分类

软件在生存期的每个阶段都可能存在错误，即计划错误、需求分析错误、设计错误、编码错误、测试错误、运行与维护错误。由于生存期的每一个阶段又是下一个阶段的先导，也就是说，前一个阶段正确的设计是保证下一个阶段设计正确的必要条件，因此，要求一旦发现某个阶段存在错误，就要尽快将其排除。

软件的错误按软件生存期阶段可分为：需求分析错误、规格说明错误、设计错误、编码错误等。

① 需求分析错误

需求分析错误是在软件需求分析阶段，分析研究用户的要求后所编写文档中出现的错误。这类错误是由于问题定义不满足用户的要求而导致的错误。

② 规格说明错误

规格说明错误是指规格说明与问题定义不一致所产生的错误，主要包括以下几点。

- 不一致性错误：规格说明中功能说明与问题定义发生矛盾。
- 冗余性错误：规格说明中某些功能说明与问题定义相比是多余的。
- 不完整性错误：规格说明中缺少某些必要的功能说明。
- 不可行错误：规格说明中有些功能要求是不可行的。
- 不可测试错误：有些功能的测试要求是不现实的。

③ 设计错误

设计错误是在设计阶段产生的错误，它使系统的设计与需求规格说明中的功能说明不相符。可细分为以下几项。

- 设计不完全错误：某些功能没有被设计或设计得不完全。
- 算法错误：算法选择不合适，主要表现为算法的基本功能不满足功能要求、算法不可行或者算法的效率不符合要求。
- 模块接口错误：模块结构不合理；模块与外部数据库的接口不一致；模块之间的接口不一致。
- 控制逻辑错误：控制流程与规格说明不一致；控制结构不合理。
- 数据结构错误：数据设计不合理；与算法不匹配；数据结构不满足规格说明要求。

④ 编码错误

编码过程中的错误是多种多样的，大体可归为以下几种：数据说明错、数据使用错、计算错、比较错、控制流错、接口错、输入/输出错以及其他的错误。

对软件错误百分比的说明如表 1-9 所示。

表 1-9　软件的错误百分比

错误类型	描述	错误百分比
软件需求错误	包括软件需求制定的不合理或不正确；需求不完全；含有逻辑错误；需求分析的文档有误	9%
功能和性能错误	包括功能和性能规定的有错误，或是遗漏了某些功能；为用户提供的信息有错误，或信息不确切；对意外的异常情况处理有误等	15%
结构错误	包括程序控制流或控制顺序有误；处理过程有误等	25%
数据错误	包括数据定义或数据结构有误；数据存储或数据操作有误等	20%
实现和编码错误	包括编码错或按键错；违背编码风格或是编码标准错误；文档有误等	10%
集成错误	包括软件内部和外部接口有误；各相关部分在时间配合、数据吞吐量等方面不协调等	8%
系统结构错误	操作系统调用错或使用错、恢复错误、诊断错误、分割及覆盖错误、引用环境错误等	2%
测试定义与测试执行错误	包括测试设计错误、测试执行错误、测试文档错误、测试用例不充分等	3%
其他类型错误		7%

5. 软件测试的工具

软件测试的工具是提高软件测试效率的重要手段，是软件理论和技术发展的重要标志。目前，应用比较广泛的软件测试工具主要有以下几种类型。

（1）测试设计工具

测试设计工具有助于准备测试输入或测试数据。测试设计工具包括逻辑设计工具和物理设计工具。逻辑设计工具涉及到说明、接口或代码逻辑，有时也叫做测试用例生成器。物理设计工具操作已有的数据或产生测试数据，如可以随机从数据库中抽取记录的工具就是物理设计工具。从说明中获取测试数据的工具就是逻辑设计工具。

（2）测试管理工具

测试管理工具是指帮助完成测试计划、跟踪测试运行结果等的工具。这类工具还包括有助于需求、设计、编码测试及缺陷跟踪的工具。

（3）静态分析工具

静态分析工具直接对代码进行分析，不需要运行代码，也不需要对代码编译链接，生成可执行文件。静态分析工具一般是对代码进行语法扫描，找出不符合编码规范的地方，根据某种质量模型评价代码的质量、生成系统的调用关系图等。

（4）动态分析工具

动态分析工具与静态分析工具不同，动态分析工具一般采用“插桩”的方式，向代码生成的可执行文件中插入一些监测代码，用来统计程序运行时的数据。其与静态分析工具最大的不同就是动态分析工具要求被测系统实际运行。

（5）覆盖测试工具

覆盖工具通过一系列的测试来测试软件被测试执行的程度。覆盖工具用于单元测试中，例如，对于安全性要求高或与安全有关的系统，则要求的覆盖程度也较高。覆盖工具还可以度量设计层次结构，如调用树结构的覆盖率。

（6）负载和性能测试工具

性能测试工具检测每个事件所需要的时间，例如，性能测试工具可以测定典型或负载条件下的响应时间。负载测试可以产生系统流量，例如产生许多代表典型情况或最大情况下的事物。这种类型的测试工具用于容量和压力测试。

（7）GUI 测试驱动和捕获/回放工具

这类测试工具可使测试自动执行，然后将测试输出结果与期望输出进行比较。此类测试工具可在任何层次中执行测试：单元测试、集成测试、系统测试或验收测试。捕获/回放工具是目前使用的测试工具中最流行的一种。

（8）基于故障的测试工具

首先给出软件的故障模型，在此故障模型下，给出基于该故障模型的软件测试工具。这是目前一种有很好发展前景的软件测试工具。随着人们对软件故障认识地不断深入，软件的故障模型也会越来越完备，并更加符合实际。基于故障的软件测试工具有三个需要研究的问题：一是故障模型的准确程度，二是测试的准确程度，三是测试的自动化程度。

1.2.2 软件测试的方法

软件的测试方法有 3 种，即试题测试法、新旧两个系统进行平行处理测试法和软件测试自动化工具测试法。

1. 试题测试法

试题测试法先建立输入数据的模型，并事先用手工求得其预计输出，然后送入模型，与新系统所得结果进行核对，从而进行测试。

此时模型数据通常利用实际上已发生的数据，但有时也可能另外编制大量测试专用数据。利用试题测试法在小规模系统中较易实现，而在大规模系统中，要用这类方法检查所有程序的全部变化非常困难，有时几乎是不可能的，在这种情况下，不能一下子完成所有的运行测试，而要用划分较细的输入模型，按各个子系统进行测试，尤其是要设法分成三种处理，即基本特殊处理、出错处理、作出相应于各个流程的数据模型。其中，基本处理最好是始终连贯地进行测试。试题要进行各种改变，直至取得满意的结果为止。

2. 新旧两个系统进行平行处理测试法

这种检查是系统最后的测试，此阶段一结束，就进入系统的全面实施。不论旧系统是手工作业还是计算机处理，本方法都通过旧系统的处理结果与新系统的结果相比较进行检查。这种平均处

理的时间视业务内容而定，短则 2～3 个月，长则半年至一年。它不仅可防止向新系统转换时产生混乱，也可以作为测试新系统可靠性的考机时间。在此期间，不只限于计算机范围的检查，还应检查手工作业和人为因素方面的问题，改正不合理的部分。

在讨论新、旧两个系统所得到的结果时，要明确新、旧系统的不同点。当结果不一致时，不能武断地认为总是新系统出错，还必须考虑是否为旧系统的错误。

对平行处理所得结果的检查应重视以下各项，如图 1-10 所示。

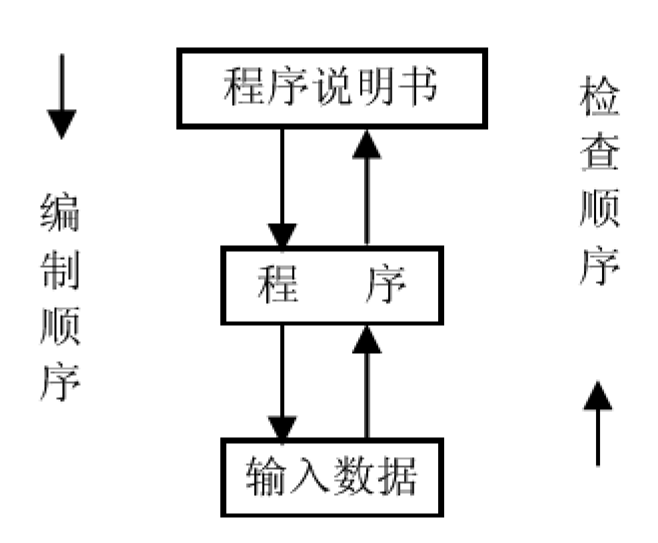


图 1-10　结果检查顺序

（1）输入数据

要注意新、旧两个系统中所用数据是否通用，做过部分修订的数据在修订处是否正确。特别是新开始用计算机处理的系统，数据的项目与旧的手工作业相比通常变化较大，所以要对代码的内容和添加、修订及删除的项目进行充分检查。新系统中所用的所有数据应是事先校验过的正确数据。

（2）程序

尽管输入数据是正确的，但如果处理数据的程序有错，仍是徒劳，因此要查看程序是否正确地满足程序说明书中所说的所有条件。此时，若分为区域、常数、输入处理、基本处理、输出处理及特殊处理后进行检查，可迅速而正确地完成任务。

（3）程序说明书

程序说明书的内容是程序设计的依据，所以必须正确完整。尤其是因机种更替而变更系统时，可作出添加、修订和删除一览表，写明旧系统程序说明书和新系统程序说明书的处理条件、处理内容、代码及项目等，突出两者的不同之处。

此外，也可考虑利用两个程序进行测试的方法：由两组程序员对同一处理分别编制完全相同的程序，然后核对两者结果。此法主要用于测试如技术计算之类的程序中找不到可比较结果的新程序。不过若两个程序的结果一致，固然可知正确，若结果不一致，就不能判断何者正确，因为缺乏基准。这种方法的费用也高，不能算是一种好的方法，但对于没有其他测试方法的业务来说，或许也只能如此。

3. 软件测试自动化工具测试法

由于测试技术的发展，出现了多种测试自动化工具，如单元测试、功能测试、黑箱测试、白箱测试、自顶向下测试、自底向上整体测试等，这些内容将在后续章节中讲解，这里不再赘述。

1.2.3 软件测试的任务

软件测试阶段有以下几方面的任务：

- 制定测试大纲。
- 制作测试数据。
- 程序测试。
- 功能测试。
- 子系统测试。
- 系统测试。
- 系统接口测试。
- 写出测试报告书。
- 制定测试大纲。



数据制作好坏，直接影响系统测试的结果，制作测试数据时应注意：由用户和程序管理组的人员制作，程序编码人员不应介入；要尽可能多的提供数据，以供检测。测试大纲是测试工作的依据，主要检查每个模块和子系统在程序设计中是否已测试过、测试的数据和输出结果是否正确、检查上一阶段交来的工作文档是否齐全、确定本阶段测试目标、制定本阶段测试内容、编写向下阶段工作提交的文档资料。

1.3 软件测试的术语定义

通过收集、加工和整编，现列出与软件测试相关的主要术语的定义，以方便软件测试人员学习及参考。

1. 80-20 原则

80%的软件缺陷常常存在于软件 20%的空间里。测试人员会根据这个原则很快找出较多的缺陷。80-20 原则的另外一种情况是：在系统分析、系统设计、系统实现阶段的复审、测试工作中能够发现和避免 80%的软件缺陷，此后的系统测试能够帮助找出剩余缺陷中的 80%，最后的 4%的软件缺陷可能只有在系统交付使用后用户经过大范围、长时间使用后才会曝露出来。实践证明 80%的软件缺陷可以借助人工测试而发现，20%的软件缺陷可以借助自动化测试得以发现。

2. α 测试

α 测试是指在软件开发人员缺席的情况下内部进行的模拟的或者实际的操作性测试。

3. β 测试

β 测试是指在软件开发人员缺席的情况下进行的操作性测试。

4. 安全性测试

安全问题日益重要，特别是对于有交互信息的网站及进行电子商务活动的网站尤其重要。安

全性的测试用于测出与系统相关的安全问题，并且能够给出安全漏洞的解决方案。

5. 白盒测试

白盒测试（白箱测试）是指基于一个应用代码的内部逻辑知识，即基于覆盖全部代码、分支、路径、条件，使用程序设计的控制结构导出测试用例。白盒测试又叫结构测试或逻辑驱动测试，它的前提是了解产品内部的工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都能按预定的要求正确工作，而不顾及它的功能，白盒测试的主要方法有逻辑驱动、基路测试等，主要用于软件验证。

6. 本地化语言测试

本地化语言测试是用于验证所有已计划要发布的不同语言版本的软件，如预期的那样被正确地翻译成当地语言。这类测试一般包括验证菜单、对话框、出错信息、帮助内容等所有用户界面上的文字都能够显示正确翻译好的当地文字。

7. 边界条件测试

边界条件测试是环绕边界值的测试。通常意味着测试软件各功能是否能正确处理最大值、最小值或者所设计软件能够处理的最长的字符串等。

8. 边界值

边界值是位于两个等价类之间的输入或输出值，或者边界附近的值。

9. 边界值覆盖

边界值覆盖是指被一组测试用例覆盖到的被测组件等价类的边界值占所有边界值的百分比。

10. 测试报告

测试报告是一份有关本次测试的总结性文档，主要记录了有关本次测试的目的、测试结果、评估结果及测试结论等信息。

11. 测试对象

测试对象是指特定环境下运行的软件系统和相关的文档。作为测试对象的软件系统可以是整个业务系统，也可以是业务系统的一个子系统或一个完整的部件。

12. 测试管理

测试管理能够建立一个完善的跟踪体系，包括报告、查询并产生报表、处理解决等。

13. 测试环境

测试环境指对软件系统进行各类测试所基于的软、硬件设备和配置。一般包括硬件环境、网络环境、操作系统环境、应用服务器平台环境、数据库环境以及各种支撑环境等。

14. 测试计划

测试计划是指对软件测试的对象、目标、要求、活动、资源及日程进行整体规划，以保证软件系统的测试能够顺利进行的计划性文档。

15. 测试免疫性（软件缺陷免疫性）

软件缺陷与病毒一样具有可怕的“免疫性”，测试人员对其采用的测试越多，其免疫能力就越强，寻找更多软件缺陷就更加困难。必须更换不同的测试方式和测试数据，尽可能地采用多种途径进行测试。

16. 测试流程

测试流程是指为了保证测试质量而精心设计的一组科学、合理、可行的有序活动。比较典型的测试流程一般包括制定测试计划、编写测试用例、执行测试、跟踪测试缺陷、编写《测试报告》等活动。

17. 测试评估

测试评估是指对测试过程中的各种测试现象和结果进行记录、分析和评价的活动。

18. 测试用例

测试用例指对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术和策略的文档；内容包括测试目标、测试环境、输入数据、测试步骤、预期结果、测试脚本等。

19. 程序代码合法性检查

程序代码合法性检查的主要标准为《intergrp 小组编程规范》，目前采用由 SCM 管理员进行规范的检查，未来期望能够有相应的工具进行测试。

20. 代码合法性测试

代码合法性测试主要包括两个部分：程序代码合法性检查与显示代码合法性检查。

21. 单元测试

单元测试是对软件中的基本组成单位进行测试，如一个模块、一个过程、一个函数或一个类等。单元测试是软件测试最基本的组成部分，也是最重要的部分之一。其目的是检验软件基本组成单位的正确性。一个软件单元的正确性是相对该单元的规约而言的，因此，单元测试是以被测单位的规约为基准。单元测试又称模块测试。

22. 动态测试和静态测试

动态测试和静态测试是一种分类方法，动态测试包含黑盒测试和白盒测试。静态测试一般只含有白盒测试。静态测试只能定性的分析软件的质量，而不能定量。从这种意义上讲，静态测试具有很大的局限性，但这并不否定静态测试的重要性，因为，就发现一个错误而言，动态测试的花费要大得多。

23. 非渐增式测试

非渐增式测试在单独测试所有构成系统的组件之前不进行任何额外测试的集成测试。

24. 分支测试

分支测试是一种针对组件的测试用例设计技术，通过分支覆盖来进行测试用例设计。

25. 峰谷测试

峰谷测试兼有容量规划 ramp-up 类型的测试和渗入测试的特征。其目标是确定从高负载（例如系统高峰时间的负载）恢复转为几乎空闲、然后攀升到高负载再降低的能力。

26. 功能测试

功能测试是指为了保证软件系统功能实现的正确性、完整性及其他特性而进行的测试。功能测试用于验证软件功能能否正常按照它的设计工作、看运行软件时的期望行为是否符合原设计。

27. 黑盒测试

黑盒测试（黑箱测试）是指着眼于程序外部结构、不考虑内部逻辑结构、针对软件界面和软件功能进行的测试。

黑盒测试也称功能测试或数据驱动测试，它是已知产品所应具有的功能，通过测试来检测每个功能是否都能正常使用，在测试时，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，测试者在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息，并且保持外部信息（如数据库或文件）的完整性。黑盒测试方法主要有等价类划分、边值分析、因果图、错误推测等，主要用于软件确认测试。测试人员通过输入他们的数据然后看输出的结果从而了解软件怎样工作。通常测试人员在进行测试时不仅使用正确的输入数据，而且还会使用具有挑战性的输入数据，以及结果可能会出错的输入数据以便了解软件怎样处理各种类型的数据。黑盒测试用于辅助白盒测试。

28. 灰盒测试

灰盒测试（灰箱测试）与黑盒测试相同，都是通过用户界面进行测试，但是测试人员已经对该软件或某种软件功能的源代码程序是怎样设计的有所了解，甚至于还读过部分源代码，因此测试人员可以有的放矢地进行某种确定的条件/功能的测试。这样做的意义在于：如果知道产品内部的设计和对产品具有深入地了解，就能够更有效地从用户界面测试它的各项性能。

29. 回归测试

回归测试是在软件维护阶段，对软件修改之后进行的测试。其目的是检验对软件进行的修改是否正确。这里，修改的正确性具有两重含义：一是指所作的修改达到了预期的目的，如错误得到改正、能够适应新的运行环境；二是指不影响软件的其他功能的正确性。

回归测试是根据修复好的缺陷再重新进行的测试，目的在于验证以前出现过但已经修复好的缺陷不再重新出现。通常确定所需要的再测试的范围时是比较困难的，所以在验证修好的缺陷时不仅要服从缺陷原来出现时的步骤重新测试，而且还要测试有可能受影响的所有功能，因此应当鼓励

对所有回归测试用例进行自动化。

30. 基本测试集

基本测试集是基于代码的逻辑结构且保证一定的分支覆盖率的测试用例的集合。

31. 基本块

基本块由一个不包含任何分支的一个或者多个连续的、可执行的指令组成的序列。

32. 基准测试

基准测试的关键是要获得一致的、可再现的结果。可再现的结果有两个好处：减少重新运行测试的次数；对测试的产品和产生的数字更为确信。使用的性能测试工具可能会对测试结果产生很大影响。假定测试的两个指标是服务器的响应时间和吞吐量，它们会受到服务器上的负载的影响。服务器上的负载受两个因素影响：同时与服务器通信的连接（或虚拟用户）的数目、每个虚拟用户请求之间的考虑时间的长短。很明显，与服务器通信的用户越多，负载就越大。同样，请求之间的考虑时间越短，负载也越大。这两个因素的不同组合会产生不同的服务器负载等级。

33. 集成测试

集成测试是指对程序模块采用一次性或增值方法组装起来，对模块间接口进行正确性检验的测试工作，集成测试又称组装测试。在软件系统集成过程中所进行的测试，其主要目的是检查软件单位之间接口的正确性。它根据集成测试计划，一边将模块或其他软件单位组合成越来越大的系统，一边运行该系统，以分析所组成的系统是否正确、各组成部分是否合拍。集成测试的策略主要有自顶向下和自底向上两种。

34. 集成与兼容性测试

集成与兼容性测试是指验证该功能能够如预期的那样与其他程序或者构件协调工作。兼容性经常意味着新、旧版本之间的协调，也包括测试的产品与其他产品的兼容使用，如使用同样产品的新版本时不影响与使用旧版本用户之间的操作。

35. 接收测试

接收测试是用来使一个用户、客户或者其他的权威机构决定是否接收一个系统或者组件的测试。

36. 可接受性测试

可接受性测试是指把测试的版本交付给测试部门大范围测试以前进行的对最基本功能的简单测试，因为在把测试的版本交付给测试部门大范围测试以前，应该先验证该版本对于所测试的功能基本上比较稳定，必须满足一些最低要求，如程序不会很容易就挂起或崩溃。如果一个新版本没通过可测试性的验证，就应该阻拦测试部门花时间在该测试版本上测试。同时还要找到造成该版本不稳定的主要缺陷并督促尽快加以修正。

37. 可靠性测试与排错性测试

可靠性测试是以验证或评估软件的可靠性为目的，并不关心测试过程中所发现的错误。软件错误是不可能完全避免的，软件存在的错误并不影响软件的交付，用户所关心的是软件的可靠性究竟是多少。排错性测试则恰恰相反，该测试是以排除软件错误为目的，一旦测试发现错误，就立刻予以排除。一般而言，排错性测试用于软件测试的早期阶段，并以白盒测试为主要测试手段。而可靠性测试用于软件测试的末尾阶段，一般以黑盒测试为主要测试手段。

38. 链接测试

链接测试对于网站用户而言意味着能不能流畅的使用整个网站提供的服务，因而链接将作为一个独立的项目进行测试。

39. 路径测试

路径测试就是设计足够的测试用例，覆盖程序中所有可能的路径。这是最强的覆盖准则。但在路径数目很大时，真正做到完全覆盖是很困难的，必须把覆盖路径数目压缩到一定限度。

40. 强力测试

强力测试通常用于验证软件的性能在各种极端的环境和系统条件下是否还能正常工作，或者说是验证软件的性能在各种极端环境和系统条件下的承受能力，如在最低的硬盘驱动器空间或系统记忆容量条件下，验证程序重复执行打开和保存一个巨大的文件 1000 次后也不会崩溃或死机。

41. 全球化测试

全球化测试的目的是检测应用程序设计中可能阻碍全球化的潜在问题，用于确保代码可以处理所有国际支持而不会破坏的功能。全球化测试使用每种可能的国际输入类型，针对任何区域性或区域设置检查产品的功能是否正常。正常的产品功能假定该组件性能稳定，能按照设计规范运行（不考虑国际环境设置或区域性/区域设置），并且数据的表示方式正确。

42. 缺陷的必然性

缺陷的必然性是指在软件测试中，由于错误的关联性，并不是所有的软件缺陷都能够得以修复，就算某些软件缺陷虽然能够得以修复，但在修复的过程中难免会引入新的软件缺陷。很多软件缺陷之间是相互矛盾的，一个矛盾的消失必然会引发另外一个矛盾的产生，如在解决通用性的缺陷后往往会带来执行效率上的缺陷。更何况在缺陷的修复过程中，常常还会受时间、成本等方面的限制，无法有效、完整地修复所有的软件缺陷，因此评估软件缺陷的重要度、影响范围、选择一个折中的方案或是从非软件的因素（如提升硬件性能）考虑软件缺陷成为在面对软件缺陷时一个必须直面的事实。

43. 确认测试

确认测试是指在模拟（或正式）的生产环境下，运用黑盒测试的方法，验证所测软件是否满足用户需求说明书中所列出的需求，确认测试又称有效性测试。

44. 人工测试

人工测试是采用人工的手段对软件实施的测试，它是相对自动测试而言的。和静态测试不同，人工测试贯穿于软件生存期的各个阶段，并通过人工运行和审查形式对软件实施的测试。在人工测试中，主要是测试人员根据一些成熟的软件设计规则对软件的正确性等进行审查，检验所进行的设计是否能满足需要。实践表明，人工测试非常有效，但往往被忽略。

45. 软件缺陷

软件缺陷的具体含义包括以下几个因素：

- 软件未达到客户需求的功能和性能。
- 软件超出客户需求的范围。
- 软件出现客户需求不能容忍的错误。
- 软件的使用未能符合客户的习惯和工作环境。

考虑到设计等方面的因素，可以认为软件缺陷还可以包括软件设计不符合规范、未能在特定的条件（资金、范围等）达到最佳等。可惜的是，很多人更倾向于把软件缺陷看成是运行时出现的问题，认为软件测试仅限于程序提交之后。

46. 渗入测试

渗入测试是一种比较简单的性能测试。渗入测试所需要的时间较长，它使用固定数目的并发用户测试系统的总体健壮性。这些测试将会通过内存泄漏、增加的垃圾收集或系统的其他问题显示因长时间运行而出现的任何性能降低的问题。测试运行的时间越久，对系统就越了解。运行两次测试是一个好主意——一次使用较低的用户负载（要在系统容量之下，以便不会出现执行队列），另一次使用较高的负载（以便出现积极的执行队列）。

47. 数据驱动测试

数据驱动测试是把测试数据写入到简单表格中，这种测试技术得到了越来越广泛的应用，这种方法被称为表驱动、数据驱动或者“第三代”自动化测试。这种方法需要编写一个解析器，用来解释表格中的数据，并执行测试。该测试架构的最主要的好处是：它允许把测试内容写在具有一定格式的表格中，用于方便数据设计和数据监视。如果测试组中有缺少编程经验的业务专家参与测试，采用数据驱动测试方法是很合适 的。数据驱动测试的解析器主要是由测试库和上层的少量开发语言编写的代码组成。

48. 探索式测试

探索式测试是不使用可识别的测试用例设计技术所进行的测试。

49. 稳定性测试

稳定性测试是指网站的运行中整个系统是否运行正常，目前没有更好的测试方案，主要采用将测试服务器长时间运转的方法进行测试。

50. 系统测试

系统测试是指将通过集成测试的软件系统或子系统，作为基于计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其他系统元素组合在一起所进行的测试工作，目的在于通过与系统的需求定义作比较，发现软件与系统定义不符合或与之矛盾的地方。对已经集成好的软件系统进行彻底的测试，以检验软件系统的正确性和性能（如运算的精度、系统的反应时间）等是否满足其规约所指定的要求。系统测试在许多情况下是比较复杂的，仅仅编制一个测试程序是不够的，往往伴随着要建立一个软硬结合的测试系统。

51. 性能测试

性能测试是指为了评估软件系统的性能状况和预测软件系统性能趋势而进行的测试和分析。性能测试通常验证软件的性能在正常环境和系统条件下重复使用是否还能满足性能指标，或者执行同样任务时新版本不比旧版本慢。一般还检查系统记忆容量在运行程序时会不会流失，如验证程序保存一个巨大的文件时，新版本不比旧版本慢。

52. 性能规划测试

性能规划测试是指对于性能规划类型的测试来说，其目标是找出在特定的环境下给定应用程序的性能可以达到何种程度。

53. 验收测试

验收测试是指在向软件的购买者展示该软件系统满足用户的要求。它的测试数据通常是系统测试数据的子集。有所不同的是，验收测试常常需要用户的代表在场，甚至在软件安装的现场。这是投入使用前的最后测试。

54. 用户界面测试

用户界面测试用于测试软件用户界面的设计是否合乎用户期望或要求。它常常包括菜单、对话框及对话框上所有按钮、文字、出错提示、帮助信息等方面的测试，如测试 Microsoft Excel 中插入符号功能所用的对话框的大小、所有按钮是否对齐、字符串的字体和大小、出错信息内容和字体大小、工具栏的位置和图标等。

55. 装配/安装/配置测试

装配/安装/配置测试是指验证软件程序在不同厂家的硬件上、所支持的不同语言的新旧版本的平台上，利用不同方式安装的软件都能够如预期的那样正确运行。

56. 自底向上的测试

自底向上的测试是指集成测试时先测试最低层的组件，然后用最低层的组件来帮助测试更高层组件的一种方法。这个过程一直重复进行，直到最高层的组件被测试到。

57. 自顶向下的测试

自顶向下的测试是指集成测试策略从软件的主控模块入手，将其直接调用的模块首先与其集

成，并将该子系统所调用的过程和所使用的数据用一些简单的代码，即模块代替，在模块的帮助下，可以运行并测试这一子系统，直至测试结果满足要求为止，然后，将这一子系统所调用的模块与该子系统集成，并重复上述过程，直到测试完毕。

1.4 软件测试的人员要求

软件测试的具体组织结构如图 1-11 所示。

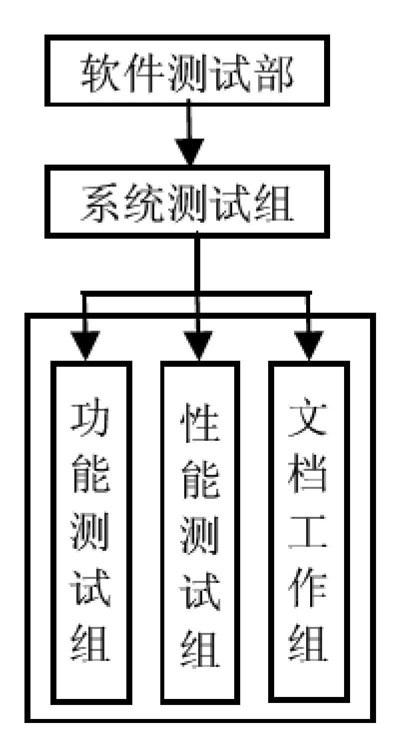


图 1-11 软件测试的具体组织结构图

由上图可以规划系统测试的人员结构。

1.4.1 系统测试人员的结构

软件测试时人员分配的概要图如图 1-12 所示。

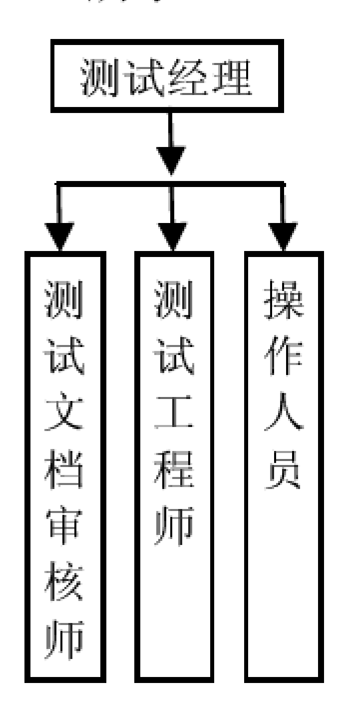


图 1-12 软件测试人员的分配图

软件测试人员最好是具有软件开发经验，理解软件工程的知识。在软件测试过程中，必须合理地组织人员。将软件测试的人员分成三部分：一部分上机测试人员（测试执行者）；一部分测试结果检查核对人员（测试工具软件开发工程师）；还有一部分是测试数据制作人员（高级软件测试工程师）。这三方面人员应该紧密配合、互相协调，以保证软件测试工作的顺利进行。

- 上机测试人员：负责理解产品的功能要求，然后根据测试规范和测试案例对其进行测试，检查软件有没有错误，决定软件是否具有稳定性，属于最低级的执行角色。

- 测试结果检查核对人员：负责编写测试工具代码，并利用测试工具对软件进行测试；或者开发测试工具为软件测试工程师服务。
- 测试数据制作人员：要具备编写程序的技术，因为不同产品的特性不一样，对测试工具的要求也是不同的，就像 Windows 的测试工具不能用于 Office，office 的测试工具也不能用于 SQL Server 一样，微软很多测试工程师就是负责专门为某个产品编写测试程序的。

在实际工作中，一般将软件测试人员定位为以下几种。

- 测试经理：测试经理主要负责测试内部管理以及与其他外部人员、客户的交流等，测试经理需要具有项目经理的知识和技能。同时测试工作开始前项目经理需要书写《测试计划书》，测试结束时需要书写《测试总结报告》。
- 测试文档审核师：测试文档审核师主要负责前置测试，包括在需求期间与设计期间产生的文档进行审核，如《需求规格说明书》、《概要设计书》、《详细设计书》等。审核需要书写审核报告。当文档确定后，需要整理文档报告，并且介绍给测试工程师。
- 测试工程师：测试工程师主要根据需求期间与设计期间产生的文档设计制作测试数据和各个测试阶段的测试用例（测试文档审核师、测试工程师可以由一组人来完成），按照测试用例完成测试工作。
- 操作人员即上机测试人员。

1.4.2　软件测试人员需要的知识

软件测试不是很快入门的职业，门槛高，需要的知识多，具有编程经验的程序员不一定是一名优秀的测试工程师。软件测试已经形成了一个独立的技术学科，软件测试技术不断更新和完善，新工具、新流程、新测试设计方法都在不断更新，没有一个合格的测试人员，测试是不可能实现高质、高效的。软件测试人员需要的知识结构如下：

- 懂得计算机的基本理论，又有一定的开发经验。
- 了解软件开发的基本过程和特征，对软件具有良好的理解能力、掌握软件测试的相关理论及技术。
- 具有软件业务经验。
- 具有根据测试计划和方案进行软件测试、针对软件需求开发测试模型、制定测试方案、安排测试计划、搭建测试环境、进行基本测试、设计简单的测试用例的能力。
- 具有规划设计环境、编制测试大纲并设计测试用例、对软件进行全面测试工作的能力。
- 具有编制测试计划、评审测试方案、规范测试流程及测试文档、分析测试结果、管理测试项目的能力。
- 能够操作软件测试工具。

1.4.3　软件测试人员需要的素质

软件测试人员需要的素质主要包括沟通能力、技术能力和较高的洞察力等。

1. 沟通能力

一名理想的测试者必须能够同测试涉及到的所有人进行沟通，即具有与技术（开发者）和非

技术人员（客户、管理人员）的交流能力。在沟通交流时一定要遵循以下要点：

- 设身处地的为客户着想，从他们的角度去测试系统。
- 考虑问题全面，即结合客户的需求、业务的流程和系统的构架等多方面考虑问题。
- 提出问题不要复杂化。
- 幽默感。

2. 技术能力

测试人员应该在开发人员研究的基础之上，更好地理解新技术并读懂程序。看懂程序可以使测试工作非常高效。不懂内部程序的人，测试工作是不可能完成的。

3. 洞察力

一个好的测试工程师应具有捕获用户观点的能力、高质量追求的愿望、对细节的关注能力。

测试人员测试的时间分配应该是：30%读程序，20%编写测试程序，50%编写测试用例和运行测试用例。

1.4.4 软件测试人员的职责

软件测试人员在测试的过程中要肩负着如下职责：

- 测试人员要了解项目需求内容，从用户角度提出自己的测试看法。
- 测试人员要编写合理的测试计划，并与项目整体计划有机地整合在一起。
- 测试人员要编写覆盖率高的测试用例。
- 测试人员要认真仔细地实施测试工作，并提交测试报告以供项目组参考。

测试人员要进行缺陷跟踪与分析。

1.5 软件测试的前景

软件测试是一门非常崭新的学科，目前处于研究探索阶段，还没有上升到理论层次。软件测试需要什么样的专业基础还没有定论，而且目前还没有一种很好的标准来衡量测试人员，但不可置疑，软件测试越来越受到软件公司的重视，软件测试工程师的作用也逐渐被人们所认可。

1. 软件测试工程师的需求

几乎每个大中型 IT 企业的软件产品在发布前都需要大量的质量控制、测试和文档工作，而这些工作必须依靠拥有娴熟技术的专业软件人才来完成。软件测试工程师就是这样的一个企业重头角色。

越来越多的 IT 企业已逐渐意识到测试环节在软件产品研发中的重要性，此类软件质量控制工作均需要拥有娴熟技术的专业软件测试人员来协作完成，软件测试工程师作为一个重头角色正成为 IT 企业招聘的热点。

由于我国企业对于软件测试自动化技术在整个软件行业中的重要作用认识较晚，因此，软件

测试工程师不足，开发测试人员比例不合理。软件测试专业技术人员在供需之间存在着巨大的缺口。有关数据显示，我国目前软件从业人才缺口高达 40 万人。业内专家预计，在未来 5～10 年中，我国社会对软件测试人才的需求数字还将继续增大。软件测试是一项需要具备较强专业技术的工作，在具体工作的过程中，测试工程师要利用测试工具按照测试方案和流程对产品进行性能测试，甚至根据需要编写不同的测试工具、设计和维护测试系统，对测试方案可能出现的问题进行分析和评估，以确保软件产品的质量。一名合格的软件测试工程师必须要经过严格的系统化职业教育培训，作为产品正式出厂前的把关人，没有专业的技术水准、没有高度的工作责任心和自信心是根本无法胜任的。

2. 软件测试工程师的发展空间

软件测试工程师未来的职业发展方向主要有以下三种：

- 走技术路线，成长为高级软件测试工程师，再向上可以成为软件测试架构设计师。
- 向管理方向发展。
- 成为开发人员，转为产品编程。

第2章　软件测试的质量要求

软件测试是软件工程的重要组成部分，测试工作的质量直接影响软件产品的生命力。本章将重点介绍以下内容：

- 软件测试的成熟度模型。
- 软件测试的流程图。
- 软件测试的流程细则。

2.1　软件测试的成熟度模型

软件测试的成熟度模型主要有三个，即 TCMM、TSM、TMM。

- TCMM: Testing Capability Maturity Model，是 1996 年由 Rodger 和 Susan Burgess 在 Testing Computer Software 会议上提出的。
- TSM: Testability Support Model，是 1996 年由 David Gelperin 和 Aldin Hayashi 提出的。
- TMM: Testing Maturity Model，是 1996 年由 IIIene Burnsein 等提出的。

TCMM 定义了 5 个级别，与 CMM 的级别设置完全相同。

- 第一级：初始级。
- 第二级：可重复级。
- 第三级：已定义级。
- 第四级：受管理级。
- 第五级：优化级。

TSM 定义了如下三个级别。

- 第一级：弱（Weak）。可测性支持，很少的测试问题被提及。
- 第二级：基本（Basic）。可测性支持，基本的测试问题被提及。
- 第三级：强（Strong）。可测性支持，所有测试问题被提及。

TSM 定义了如下 6 个 KSA（Key Support Area）。

- 测试友好的基础构架。
- 顾及测试的项目计划。
- 测试友好的产品信息。
- 顾及测试的软件设计。
- 测试件。

- 测试环境设计。

TSM 从测试组织的外部而非内部本身来考察测试的成熟性，这是它区别于另外两个测试模型的地方。

TMM 定义了如下 5 个级别。

- 第一级：初始级。
- 第二级：定义级。
- 第三级：集成级。
- 第四级：管理和测量级。
- 第五级：优化预防缺陷和质量控制级。

以上三个模型各有侧重，当前流行的是 TMM，因此本节将重点介绍 TMM。

1. TMM 的模型框架

TMM 定义了 5 个成熟度等级，每个等级代表着一个成熟的测试过程。TMM 模型框架如图 2-1 所示。

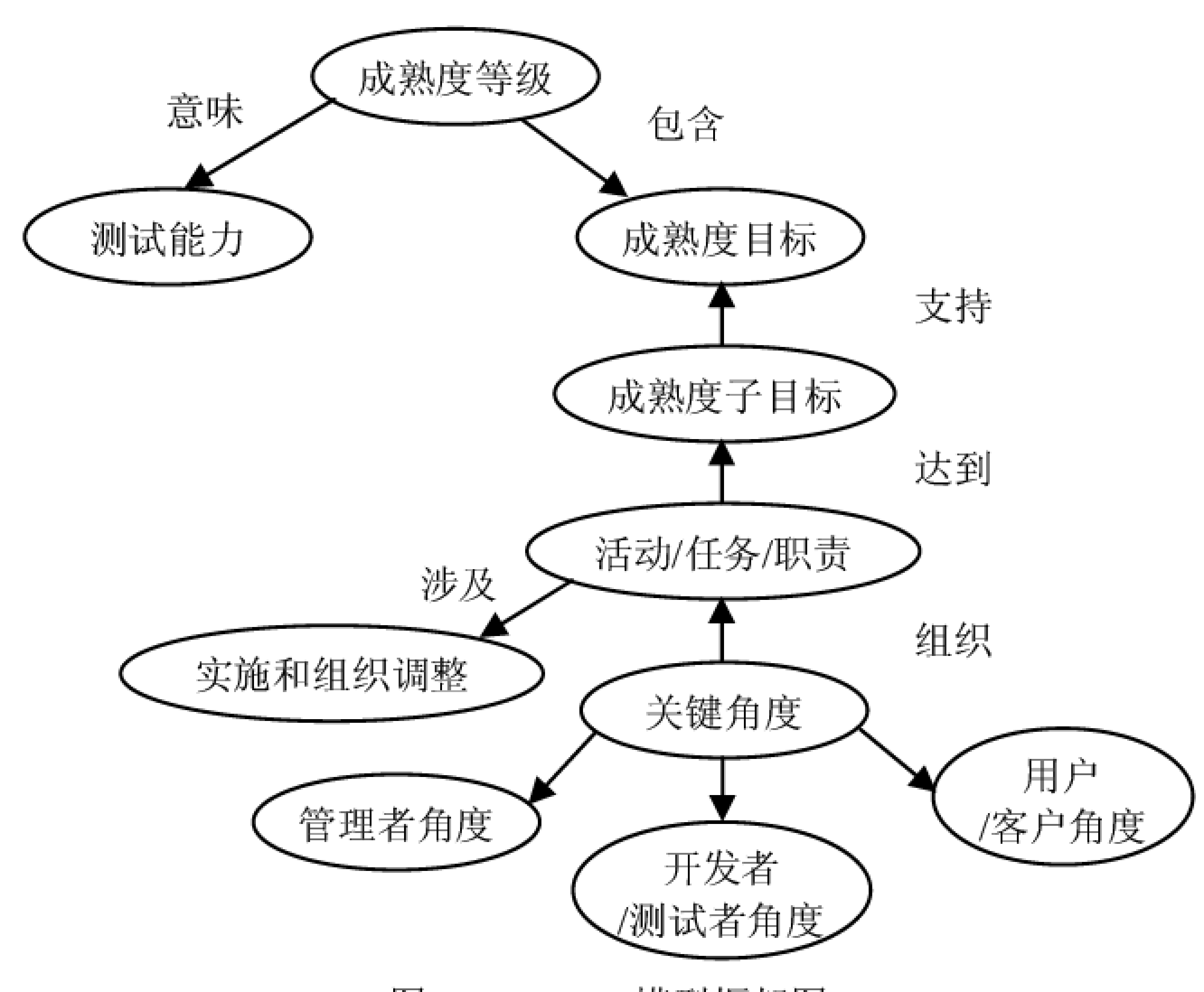


图 2-1 TMM 模型框架图

对图 2-1 的具体说明如下：

- 成熟度目标定义了达到该等级必须实现的测试改进目标。
- 成熟度子目标更为具体，定义了该等级的范围、界限和需要完成的事项。要达到某个成熟度等级，组织必须满足这个等级的成熟度目标，可通过活动/任务/职责来达到成熟度子目标。
- 活动/任务/职责涉及实施和组织调整问题。活动和任务定义了如果要改进测试能力达到某个等级所要做出的行动，它们与组织的承诺有关。
- 模型中为三组人分配了职责，这三组人是测试过程中的关键参与者，包括管理者（包括承诺及完成改进与测试过程成熟度相关的活动和任务的能力）、开发者/测试者（包括技术上

的活动和任务，这些活动和任务来自于成熟的测试实践）、用户/客户（定义为一个协作或支持角度）等，模型中称为关键角度。

2. 每个等级的成熟度目标

对每个等级的成熟度目标的说明如图 2-2 所示。

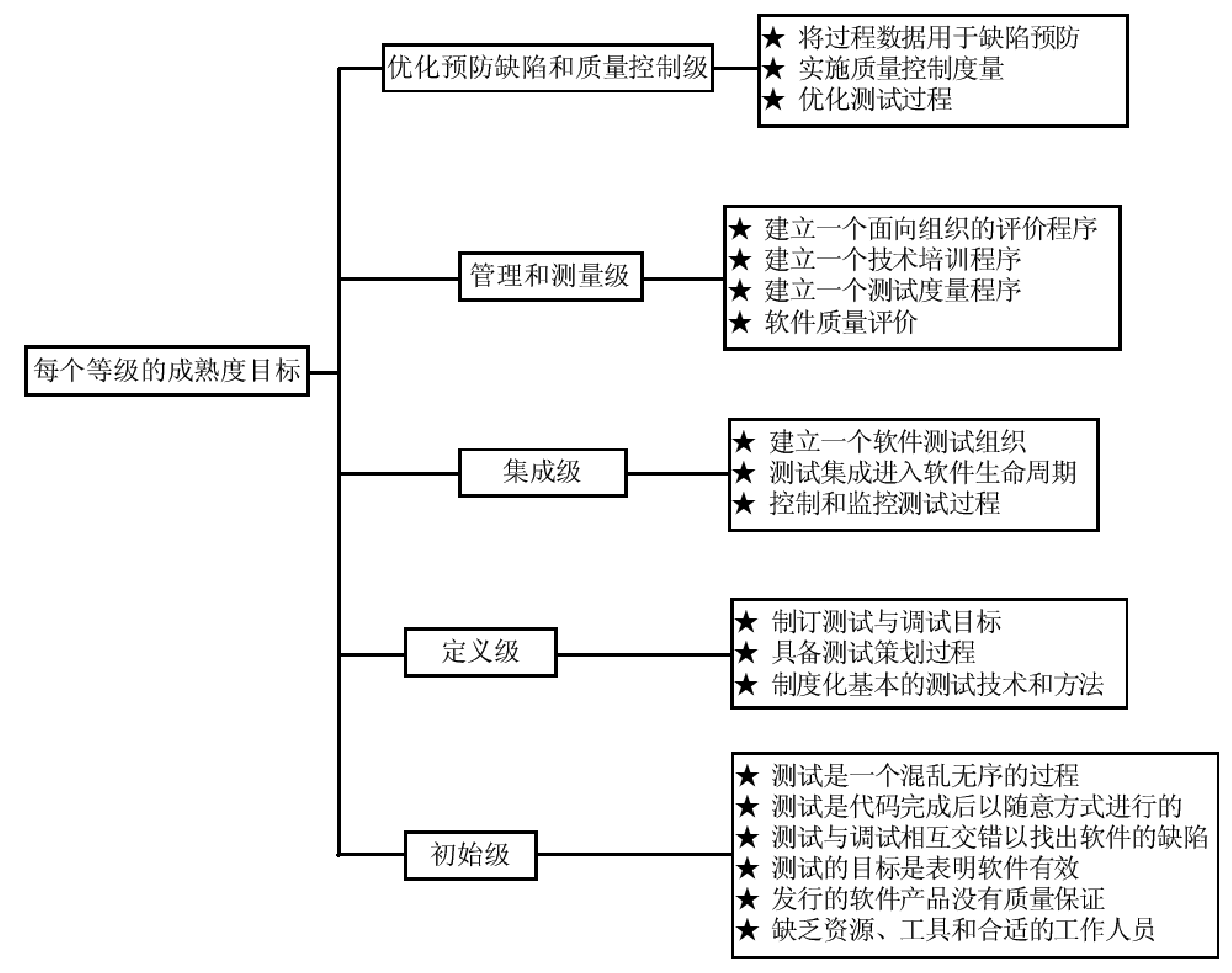


图 2-2　每个等级的成熟度目标

（1）初始级

测试是一个混乱无序的过程，几乎没有定义并且与调试没有区别。软件开发过程中缺乏测试资源、工具以及训练有素的测试人员。测试是代码完成后以随意方式进行的。测试与调试相互交错以找出软件的缺陷。

（2）定义级

在 TMM 的定义级中，测试已具备基本的测试技术和方法，软件的测试与调试已经明确地被区分开。具有以下几个目标。

① 制订测试与调试目标

测试和调试具有明显区别，分别为这两个活动定义目标、任务、活动和工具，并分配职责。正确区分这两个过程是提高软件组织测试能力的基础。与调试工作不同，测试工作是一种有计划的

活动，可以进行管理和控制。调试相关的活动经常导致过程不可预见，项目经理必须安排缺陷定位、修复、重新测试的时间和资源。这种管理和控制活动需要制订相应的策略和政策，以确定和协调这两个过程。制订测试与调试目标包含 5 个子成熟度目标：

- 分别形成测试组织和调试组织，并有经费支持。
- 规划并记录测试目标。
- 规划并记录调试目标。
- 将测试和调试目标形成文档，并分发至项目组。
- 将测试目标反映在测试计划中。

② 具备测试策划过程

制订计划是使一个过程可重复、可定义和可管理的基础。测试计划应包括测试目的、风险分析、测试策略、测试设计、规格说明和测试用例。此外，测试计划还应说明如何分配测试资源，以及如何划分单元测试、集成测试、系统测试和验收测试的任务。还需要包括测试完成准则和测试活动的所有资源、进度、职责。启动测试计划过程包含以下 5 个子目标：

- 建立组织内的测试计划组织并予以经费支持。
- 建立组织内的测试计划政策框架并予以管理上的支持。
- 开发测试计划模板并分发至项目的管理者和开发者。
- 建立一种机制，使用户需求成为测试计划的依据之一。
- 评价、推荐和获得基本的计划工具并从管理上支持工具的使用。

③ 制度化基本的测试技术和方法

为改进测试过程能力，组织中需要应用基本的测试技术和方法，并说明何时和怎样使用这些技术、方法和支持工具。必须在整个组织中实施基本的测试技术和方法，要清晰规定如何、何时实施，以及基本的支持工具。基本的测试技术和方法例子包括：黑盒、白盒测试策略；区分单元、集成、系统、验收测试。将制度化基本的测试技术和方法具有以下两个子目标：

- 在组织范围内成立测试技术组，研究、评价和推荐基本的测试技术和测试方法，推荐支持这些技术与方法的基本工具。
- 制定管理方针以保证在全组织范围内一致使用所推荐的技术和方法。

（3）集成级

在集成级，测试不仅仅是跟随在编码阶段之后的一个阶段，它已被扩展成与软件生命周期融为一体的一组已定义的活动。测试活动遵循软件生命周期的模型。测试人员在需求分析阶段便开始着手制订测试计划，并根据用户或客户需求建立测试目标，同时设计测试用例并制定测试通过准则。在集成级上，应成立软件测试组织，提供测试技术培训，关键的测试活动应有相应的测试工具予以支持。在该测试成熟度等级上，没有正式的评审程序，没有建立质量过程和产品属性的测试度量。集成级要实现 3 个成熟度目标，分别建立软件测试组织、测试集成进入软件生命周期、控制和监控测试过程。

① 建立软件测试组织

软件测试的过程及质量对软件产品的质量具有直接影响。由于测试往往是在时间紧、压力大的情况下所完成的一系列复杂的活动，因此应由训练有素的专业人员组成测试组。测试组要完成与测试有关的多种活动，包括负责制订测试计划、实施测试执行、记录测试结果、制订与测试有关的标准和测试度量、建立测试数据库、测试重用、测试跟踪以及测试评价等。建立软件测试组织需要实现以下 4 个子目标：

- 建立全组织范围内的测试组，并得到上级管理层的领导和各方面的支持，包括经费支持。
- 定义测试组的作用和职责。
- 由训练有素的人员组成测试组。
- 建立与用户或客户的联系，收集他们对测试的需求和建议。

② 测试集成进入软件生命周期

提高测试成熟度和改善软件产品质量都要求将测试工作与软件生命周期中的各个阶段联系起来，测试活动与软件生命周期的所有阶段并行开展，而并非独立进行，这对测试过程成熟度和产品质量至关重要。集成的体现包括在生命周期的早期开始进行测试策划、在生命周期的不同阶段，通过多种渠道邀请用户参与测试过程。该目标具有以下 4 个子目标：

- 将测试阶段划分为子阶段，并与软件生命周期的各阶段相联系。
- 基于已定义的测试子阶段，采用软件生命周期模型。
- 制订与测试相关的工作产品的标准。
- 建立测试人员与开发人员共同工作的机制。这种机制有利于促进将测试活动集成于软件生命周期中。

③ 控制和监视测试过程

为控制和监视测试过程，软件组织需要采取相应的措施，如制订测试产品的标准、制订与测试相关的偶发事件的处理预案、监督和控制测试过程提供的与之相关活动的可视性、确保测试过程能依据策划进行。通过与测试策划对比实际的测试工作产品、投入工作量、成本和进度来体现测试进展。控制和监督的支持包括：测试产品所用标准、测试日志、测试相关的风险应急计划、可用于评价测试进展和效率的测试度量数据，以及测试相关项的配置管理。控制和监视测试过程具有以下 3 个子目标：

- 制订控制和监视测试过程的机制和政策。
- 定义、记录并分配一组与测试过程相关的基本测量。
- 开发、记录并文档化一组纠偏措施和偶发事件的处理预案，以备实际测试严重偏离计划时使用。

（4）管理和测量级

在管理和测量级，测试活动是完全被管理的。在管理层测试者们定义、收集、分析和使用测试相关的度量数据。测试活动的定义正式扩展到整个生命周期中的审查活动，同行评审和审查作为基于实现的测试活动的补充，它们被认为是质量控制程序，用以移除软件产品的缺陷。

测试活动除测试被测程序外，还包括软件生命周期中各个阶段的评审、审查和追查，使测试活动涵盖了软件验证和软件确认活动。根据管理和测量级的要求，软件工作产品以及与测试相关的工作产品，如测试计划、测试设计和测试步骤都要经过评审，因为测试是一个可以量化并度量的过程。为了测量测试过程，测试人员应建立测试数据库、收集和记录各软件工程项目中使用的测试用例、记录缺陷并按缺陷的严重程度划分等级。管理和测量级具有 4 个成熟度目标，分别是建立面向组织的评价程序、建立技术培训程序、建立测试度量程序、软件质量评价。

① 建立面向组织的评价程序

软件组织应在软件生命周期的各阶段实施评审，以便尽早有效地识别、分类和消除软件中的缺陷。同行评审（包括审查和走查两种形式）被认为是测试活动，在生命周期所有阶段中实施同行评审，更早、更有效地识别、记录、消除软件工作产品和测试工作产品中的缺陷。建立评审程序具有以下 4 个子目标：

- 管理层要制订评审政策并支持评审过程。
- 测试组和软件质量保证组要确定并文档化整个软件生命周期中的评审目标、评审计划、评审步骤以及评审记录机制。
- 评审项由上层组织指定，通过培训参加评审的人员，使他们理解和遵循相关的评审政策、评审步骤。

② 建立技术培训程序

为高效率地完成测试工作，测试人员必须经过适当的培训。通过专业培训程序来确保为测试组提供具备技能的人员。专业培训内容应包括：测试策划；测试方法、标准、技术和工具；审查过程；如何支持用户参与测试和评审过程等。建立技术培训程序具有如下 3 个子目标：

- 制订组织的培训计划，并在管理上提供包括经费在内的支持。
- 制订培训目标和具体的培训计划。
- 成立培训组，配备相应的工具、设备和教材。

③ 建立测试度量程序

测试过程的测量程序是评价测试过程质量、改进测试过程的基础，对监视和控制测试过程至关重要，必须谨慎策划和管理测量程序，程序中应识别收集哪些测试数据，并决定由谁、如何来使用这些数据。测量包括测试进展、测试费用、软件错误、缺陷数据以及产品测量等。建立测量程序具有以下 3 个子目标：

- 定义组织范围内的测试过程的测量政策和目标。
- 制订测试过程的测量计划，测量计划中应给出收集、分析和应用测量数据的方法。
- 应用测量结果制订测试过程并改进计划。

④ 软件质量评价

软件质量评价的目的之一是判断测试过程的充分性。软件质量评价需要组织为每种类型的软件工作产品、定义质量属性和质量目标。质量目标与测试过程的充分性密切相关，因为成熟的测试过程应能保证软件产品可靠、可用、可维护、可移植和安全。软件质量评价的内容包括定义可测量

的软件质量属性、定义评价软件工作产品的质量目标等工作。软件质量评价具有以下两个子目标：

- 管理层测试组和软件质量保证组要制订与质量有关的政策、质量目标和软件产品质量属性。
- 测试过程应是结构化、已测量和已评价的，以保证达到质量目标。

（5）优化预防缺陷和质量控制级

测试首先要保证软件产品满足规格说明书，并对它的可靠性有信心，其次，测试要处理缺陷和预防缺陷，这点可通过收集和分析缺陷数据来实现。由于本级的测试活动是可重复、已管理、已定义和已测量的，因此软件组织能够优化调整和持续改进测试过程。测试过程的管理为持续改进产品质量和过程质量提供指导，并提供必要的基础设施。优化预防缺陷和质量控制级具有 3 个子目标，包括将过程数据用于缺陷预防、实施质量控制度量、优化测试过程。

① 将过程数据用于缺陷预防

这时的软件组织能够记录软件缺陷、分析缺陷模式、识别错误根源、制订防止缺陷再次发生的计划、提供跟踪活动的办法，并将这些活动贯穿于全组织的各个项目中。将过程数据用于缺陷预防具有如下 4 个子目标：

- 成立缺陷预防组。
- 识别和记录在软件生命周期的各阶段中引入的软件缺陷和消除的缺陷。
- 建立缺陷原因分析机制，确定缺陷原因。
- 管理、开发和测试人员互相配合制订缺陷预防计划，防止已识别的缺陷再次发生。缺陷预防计划要具有可跟踪性。

② 实施质量控制度量

软件组织通过采用统计采样技术、测量组织的自信度、测量用户对组织的信赖度以及设定软件可靠性目标来推进测试过程。为了加强软件质量控制，测试组和质量保证组要有负责质量的人员参加，他们应掌握能减少软件缺陷和改进软件质量的技术和工具。实施质量控制度量具有如下 4 个子目标：

- 软件测试组和软件质量保证组建立软件产品的质量目标，如产品的缺陷密度、组织的自信度以及可信赖度等。
- 测试管理者要将这些质量目标纳入测试计划中。
- 培训测试组学习和使用统计学方法。
- 收集用户需求以建立使用模型。

③ 优化测试过程

在测试成熟度的最高级，已能够量化测试过程。这样就可以依据量化结果来调整测试过程、不断提高测试过程能力，并且软件组织具有支持这种能力持续增长的基础设施。基础设施包括政策、标准、培训、设备、工具以及组织结构等。优化测试过程包含如下 5 个子目标：

- 识别需要改进的测试活动。
- 实施改进。
- 跟踪改进进程。

- 不断评估所采用的与测试相关的新工具和新方法。
- 支持技术更新。

2.2　软件测试的流程图

实际上软件测试是由确认、验证和测试三个方面组成的。

- 确认：用于评估将要开发的软件产品是否正确无误、可行和有价值。
- 验证：用于检测软件开发的每个阶段、每个步骤的结果是否正确无误、是否与软件开发各阶段的要求或期望的结果相一致。
- 测试：通常包括单元测试、集成测试、系统测试等。

软件测试的流程图可分为测试项目整体流程图、测试项目确认流程图、测试执行流程图、测试策划流程图、问题跟踪与测试关闭流程图、嵌入式软件测试流程图、软件整体测试流程图。

1. 测试项目整体流程图

测试项目整体流程如图 2-3 所示。

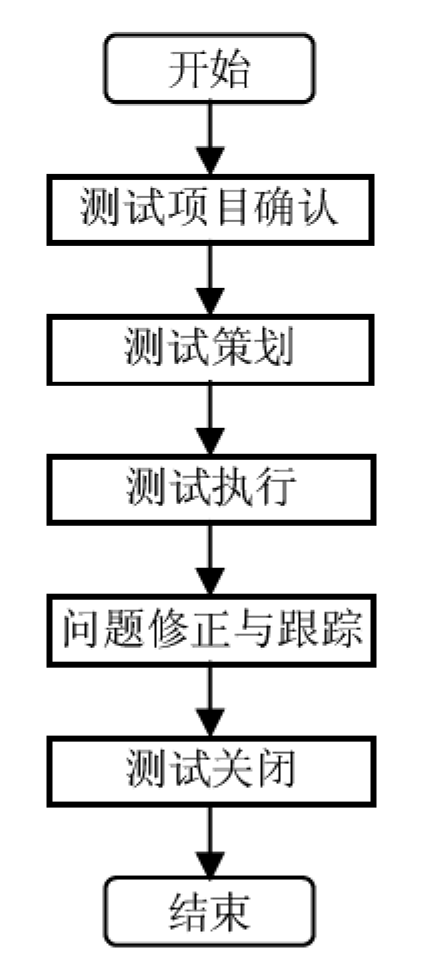


图 2-3　测试项目整体流程图

2. 测试项目确认流程图

测试项目确认流程如图 2-4 所示。

3. 测试执行流程图

测试执行流程如图 2-5 所示。

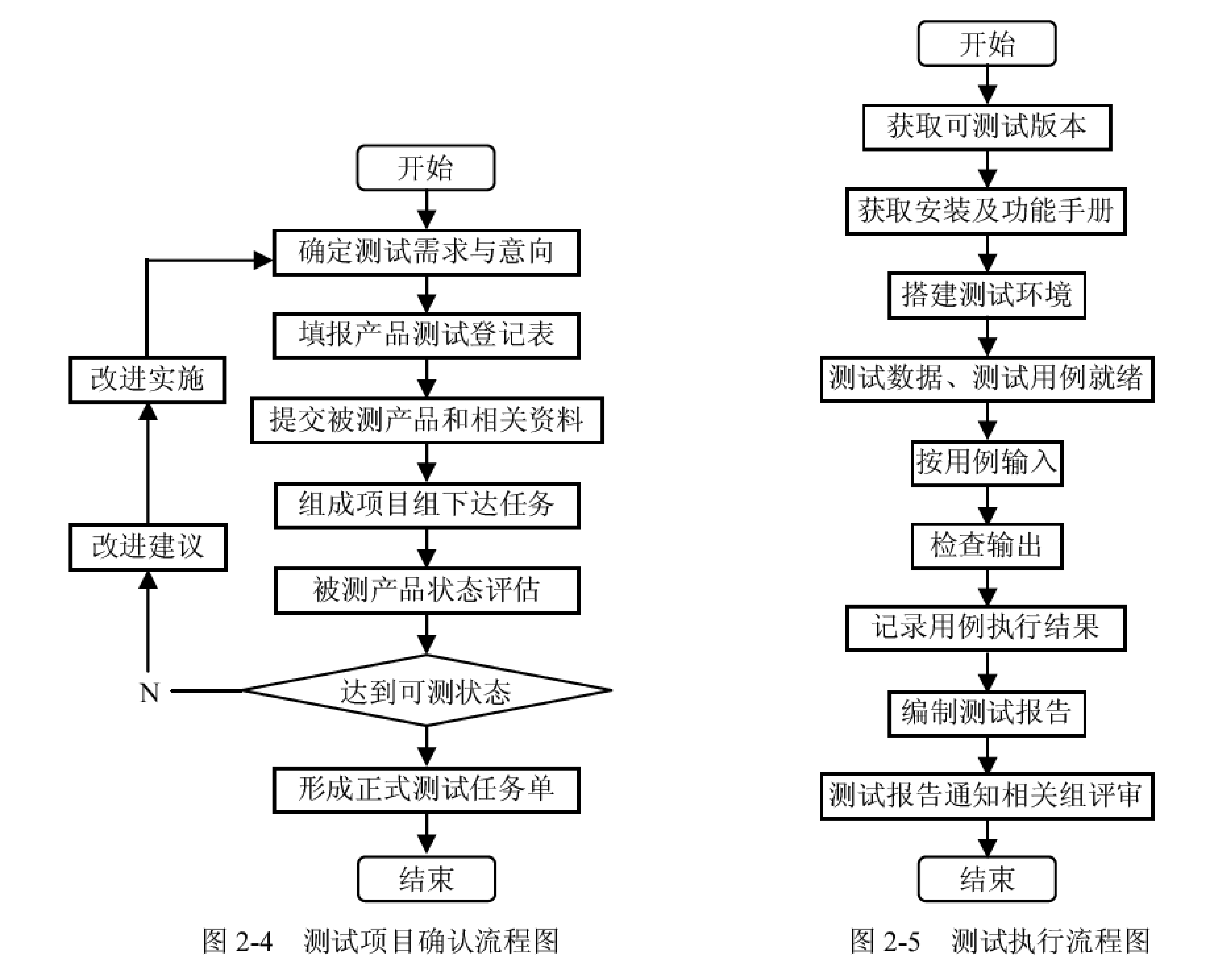


图 2-4　测试项目确认流程图

图 2-5　测试执行流程图

4. 测试策划流程图

测试策划流程如图 2-6 所示。

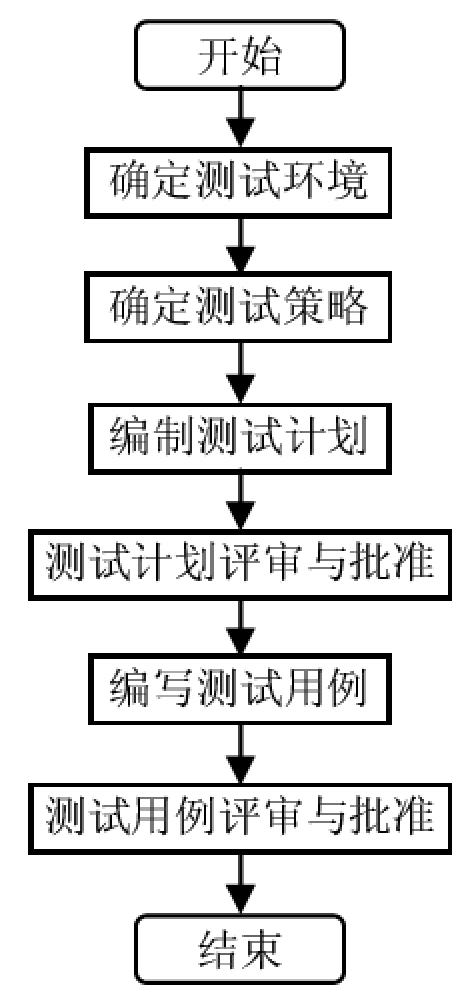


图 2-6　测试策划流程图

5. 问题跟踪与测试关闭流程图

问题跟踪与测试关闭流程如图 2-7 所示。

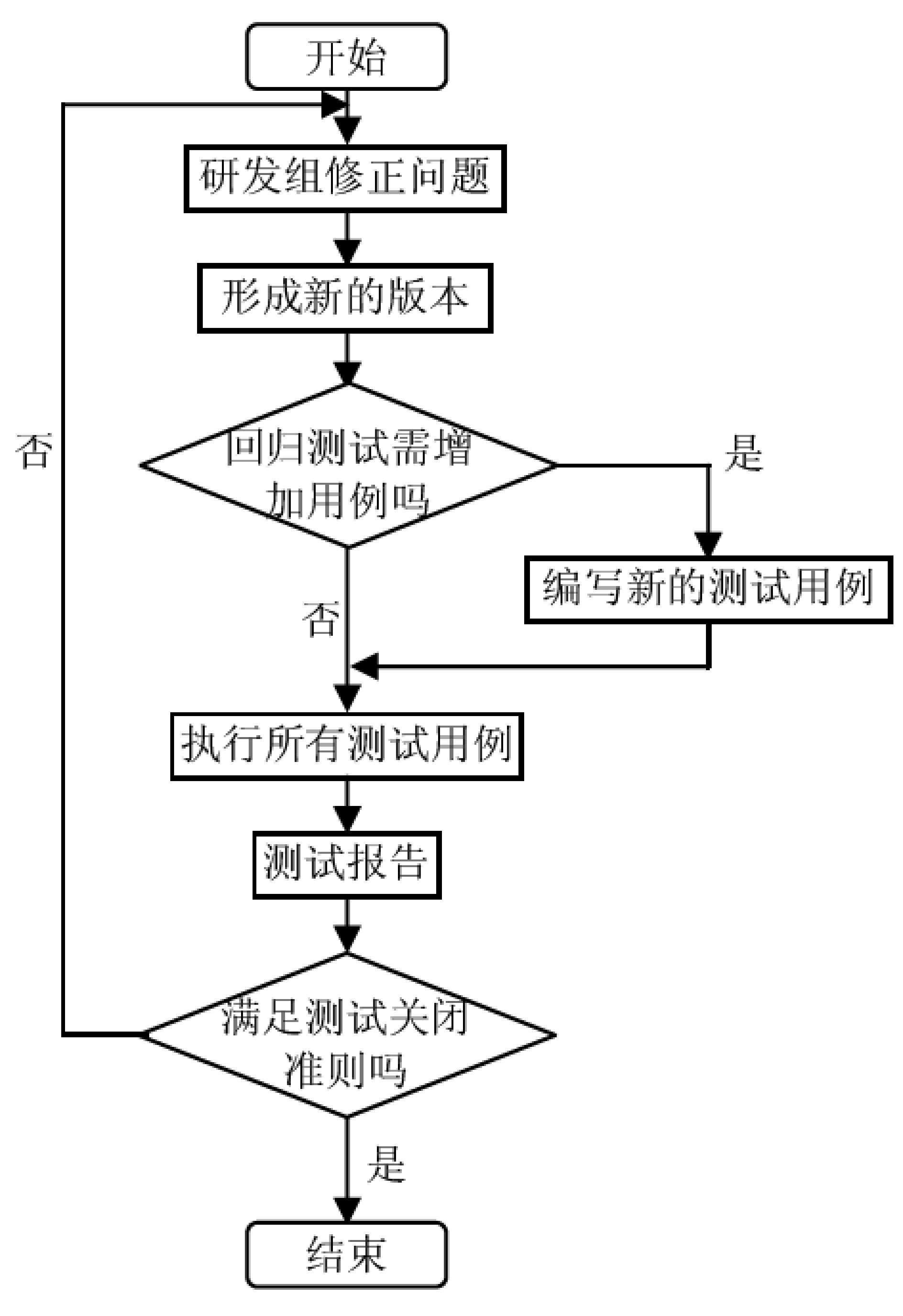


图 2-7　问题跟踪与测试关闭流程图

6. 嵌入式软件测试流程图

嵌入式软件测试流程如图 2-8 所示。

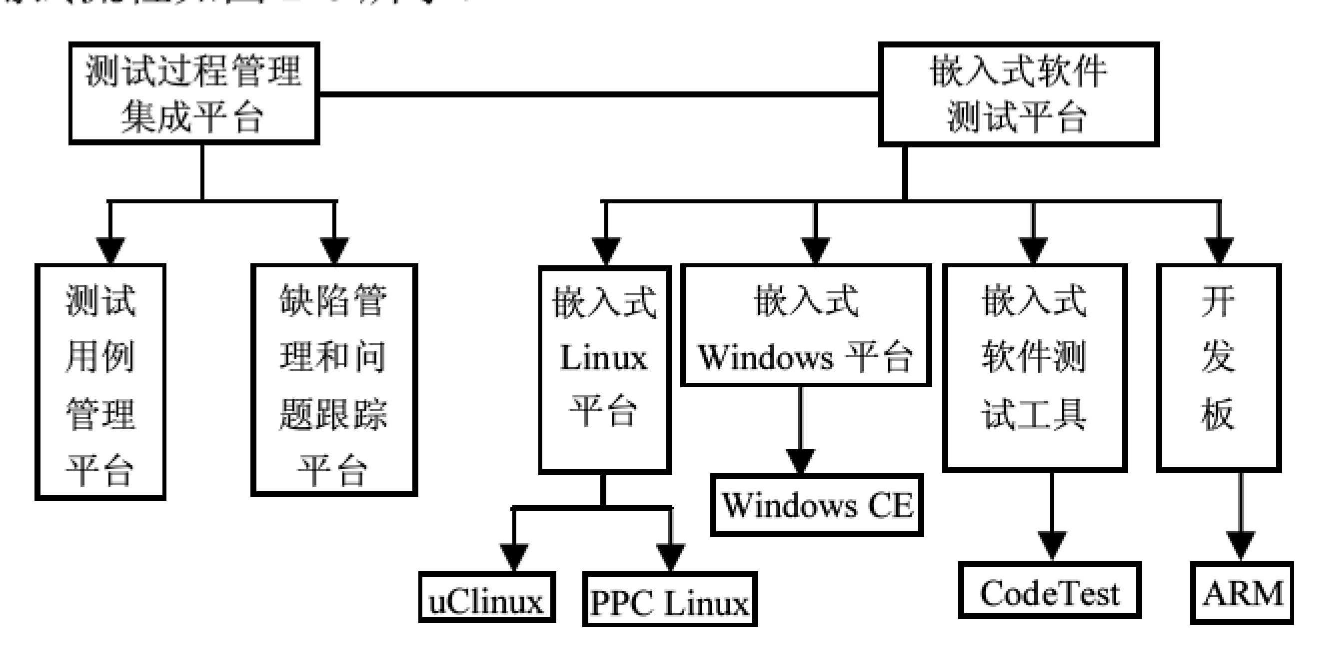


图 2-8　嵌入式软件测试流程图

7. 软件整体测试流程图

软件整体测试流程如图 2-9 所示。

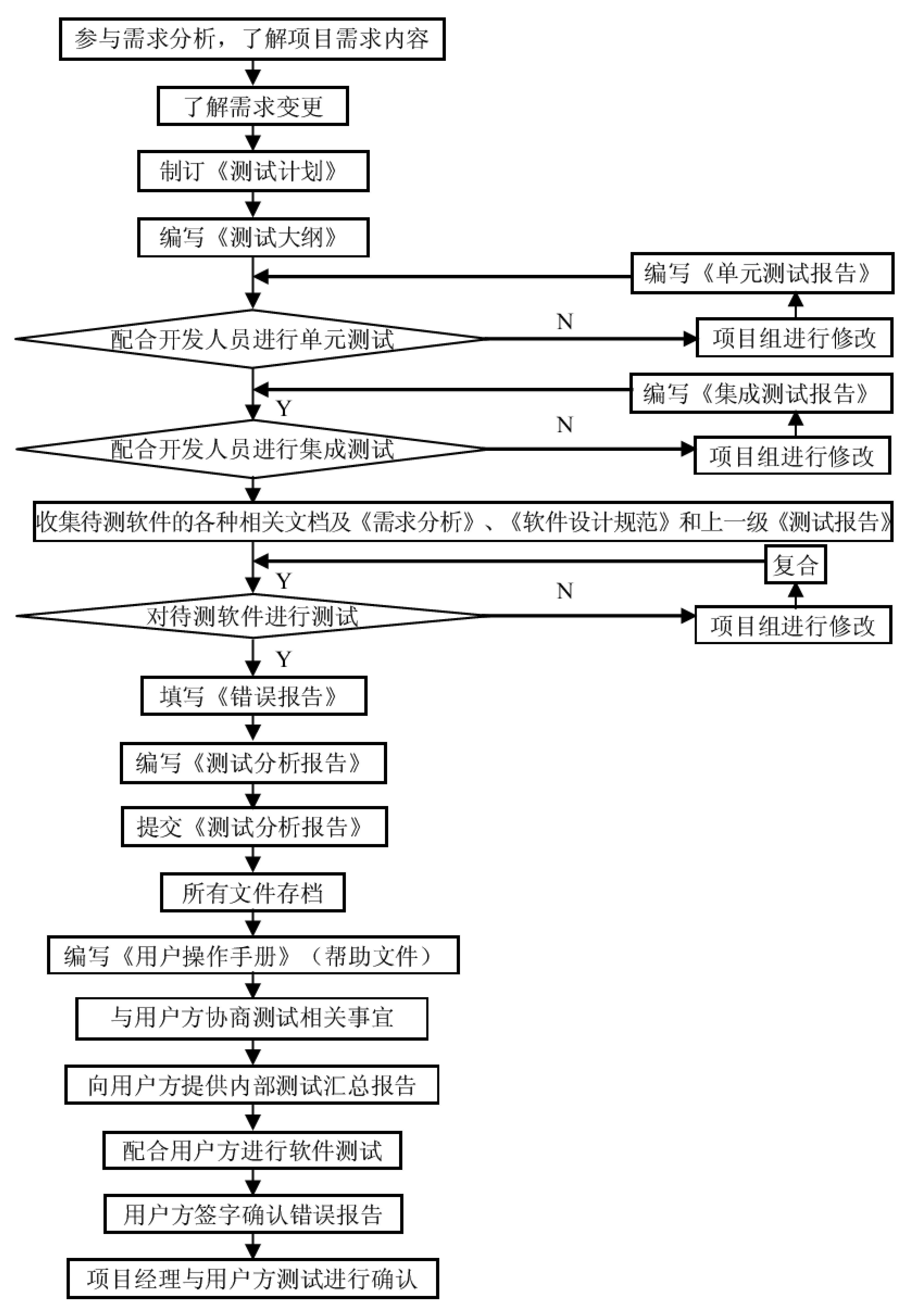


图 2-9　软件整体测试流程图

2.3　软件测试的流程细则

软件测试的流程细则主要从需求阶段、设计编码阶段、测试阶段、用户测试阶段等方面进行介绍。

1. 需求阶段

需求阶段具有如下要求：

- 测试人员了解项目需求并收集结果，包括项目需求规格说明、功能结构及模块划分等。
- 测试人员了解项目需求变更。
- 测试人员会同项目主管根据软件需求制定和确定测试进度时，必须由开发人员和相关的测试部门人员共同进行。在制定测试进度时，必须考虑到合理地配置测试资源（测试设备、测试所要用的技术文档资料、测试人员和对人员进行必要的培训）。
- 为了使所制定的测试进度正常有效，必须对其所制定的测试进度加以量化。要制定测试的各个阶段的测试进度。有特殊情况时还必须制定特定系统的测试进度，如文件管理系统、资料库内容功能测试等。
- 所制定的测试进度中必须含有修改问题和复查的时间。

2. 设计编码阶段

设计编码阶段具有如下要求：

- 测试人员制定测试大纲、测试设计、测试用例。
- 对每一个测试需求确定其需要的测试用例。
- 对每一个测试用例确定其输入及预期结果。
- 确定测试用例的测试环境配置、需要的驱动界面或稳定性。
- 为测试用例准备输入数据。
- 编写《测试用例文档》。
- 对测试用例进行同行评审。
- 项目开发组对完成的功能模块进行单元测试，测试人员参与单元测试过程；单元测试完成，编写《单元测试报告》。
- 所有单元测试及相应的修改完成后，项目开发组组织进行确认测试和系统集成测试，测试人员参与集成测试过程；集成测试完成后，编写《集成测试报告》。

3. 测试阶段

测试阶段具有如下要求：

- 项目开发组完成集成测试后，提交测试所要求的待测软件及各种文档、手册、前期测试报告。
- 测试组安排和协调测试设备、环境等准备工作。
- 测试组按测试计划、测试大纲的要求对待测软件进行有效性测试、集成测试。
- 填写《错误报告》。
- 对修改后的情况进行复核。
- 测试结束后，测试人员对测试结果进行汇总；测试主管审核测试结果，得出测试结论；测试组进行测试分析和评估，编写《测试分析报告》。
- 提交《测试分析报告》。
- 制作《用户操作手册》。

4. 用户测试阶段

用户测试阶段具有如下要求：

- 项目开发组与用户方商定测试计划、测试内容、测试环境等。
- 项目测试组向用户方提供项目内部《测试总结报告》。
- 由项目开发组或测试组配合用户进行用户方测试。
- 由用户方编制《用户方软件测试报告》（包括《程序错误报告》和《测试分析报告》），若用户方无法编制测试报告，则经与用户方协商由项目开发组编制《用户方软件测试报告》，经用户方签字后即可生效。
- 项目经理与用户方对用户方测试进行确认。

第3章　白盒测试技术

白盒测试技术是软件测试的主要方法之一，白盒测试的基本概念、依据、流程、方法、工具、测试用例等内容是必须掌握的，本章重点讨论以下内容：

- 白盒测试的基本概念。
- 白盒测试的依据和流程。
- 白盒测试的方法。
- 白盒测试的要求。
- 白盒测试的工具。

3.1　白盒测试的基本概念

白盒测试（白箱测试）是指基于一个应用代码的内部逻辑知识，即基于覆盖全部代码、分支、路径、条件，使用程序设计的控制结构导出测试用例，是软件测试的主要方法之一。这种测试对程序的要求很高，需要了解程序的构架、具体需求，以及一些编写程序的技巧，能够检查一些程序规范、指针、变量、数组越界等问题，使得问题在前期就暴露出来。白盒测试又叫结构测试或逻辑驱动测试，它是在了解产品内部工作过程的基础上进行的，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都能按预定要求正确工作，而不顾它的功能，白盒测试的主要方法有逻辑驱动、基路测试等，主要用于软件验证。一般程序容易犯的错误包括：没有定义变量、无效引用、野指针、超过数组下标、内存分配后没有删除、无法调入循环体、函数本身没有析构、导致循环失效或者死循环、参数类型不匹配、调用系统的函数没有考虑到系统的兼容性等。

白盒测试一般是以单元或者模块为基础。目前的做法是把它归结为开发的范畴，请专人对代码进行分析或者利用部分工具（如 Rational 系列、Boundchecker 等工具）协助发现变量没有初始化、指针错误等问题，从而大大减少了人力。

白盒测试方法必须遵循以下 4 条原则才能达到测试的目的：

- 保证一个模块中的所有独立路径至少被测试一次。
- 所有逻辑值均需要测试真和假两种情况。
- 检查程序的内部数据结构，保证其结构的有效性。
- 在上、下边界及可操作范围内运行所有循环。

测试具体实施的内容分为以下 8 大类：

- 软件各层公用问题的测试。
- Java 语言的测试。
- 数据类型的测试。
- SQL 语句的测试。
- 界面（UI）的测试。
- 数值对象（VO）的测试。
- 业务对象（BO）的测试。
- 数据管理对象（DMO）的测试。

白盒测试中测试的策略是：首先进行静态结构分析，采用先静态后动态的组合方式，然后进行覆盖测试。利用静态结构分析的结果，通过代码检查和动态测试的方法对结果进行进一步确认，使测试工作更为有效。

白盒测试在不同的测试阶段的侧重点如下。

- 单元测试：代码检查、逻辑覆盖。
- 集成测试：增加静态结构分析、静态质量度量。
- 系统测试：根据黑盒测试结果，采用白盒测试。

3.2　白盒测试的依据和流程

1. 白盒测试的依据

白盒测试的主要依据如下：

- 软件产品的需求报告。
- 软件产品的需求规格说明书。
- 软件产品的设计文档。
- 软件产品的界面。
- 软件产品的编码规范。
- 软件产品的开发命名标准。
- 软件产品设计文档的相关规范。

2. 白盒测试的流程

测试流程可分为界面对象和业务对象两种方式。

（1）　界面对象测试流程

界面对象测试流程的示意图如图 3-1 所示。

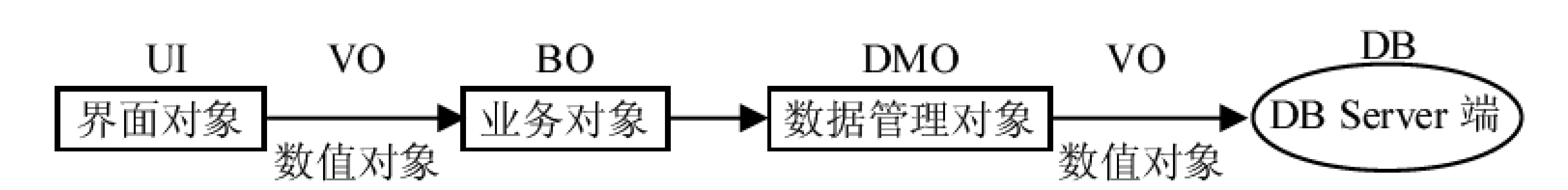


图 3-1　界面对象测试流程图

界面对象的优点：便于测试者从界面层直观地录入数据。

界面对象的缺点：进行回归测试时，需要重复录入数据。

（2）业务对象测试流程

业务对象测试流程的示意图如图 3-2 所示。

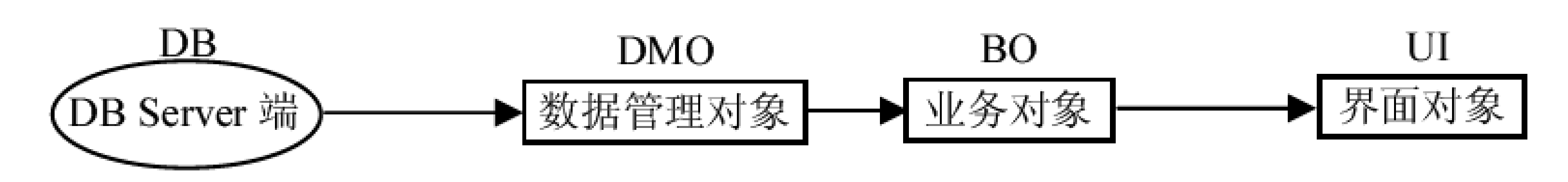


图 3-2　业务对象测试流程图

业务对象是从底层开始测试，底层测试通过了，再依次向上一层测试。

业务对象的优点：进行回归测试时，不需要再输入数据，执行一遍测试程序即可。

业务对象的缺点：需要给中间层编写一个测试小程序，即根据程序中类的对象构造输入数据及将结果输出到控制台上。

3.3　白盒测试的方法

白盒测试的优点是帮助软件测试人员增大代码的覆盖率、提高代码的质量、发现代码中隐藏的问题。白盒测试主要用于检查程序的内部结构、逻辑、循环和路径。常用的测试用例设计方法有代码检查法、静态结构分析法、静态质量度量法、逻辑覆盖法、基本路径测试法、域测试法、符号测试法、Z 路径覆盖法、程序变异测试法等，运用最为广泛的是基本路径测试法。

3.3.1　代码检查法

代码检查可以发现的软件问题包括以下几项：

- 检查代码和设计的一致性。
- 代码对标准的遵循、可读性。
- 代码逻辑表达的正确性。
- 代码结构的合理性。
- 程序编写与编写标准的符合性。
- 程序中不安全、不明确和模糊的部分。
- 编程风格问题等。

检查代码时需要进行检查的内容如下。

- 检查变量的交叉引用表：检查未说明的变量和违反了类型规定的变量、变量的引用和使用情况。

- 检查标号的交叉引用表：验证所有标号的正确性。
- 检查子程序、宏、函数：验证每次调用与所调用位置是否正确，调用的子程序、宏、函数是否存在，参数是否一致。
- 等价性检查：检查全部等价变量的类型的一致性。
- 常量检查：确认常量的取值、数制、数据类型。
- 标准检查：检查程序中是否违反标准的问题。
- 风格检查：检查程序的设计风格。
- 比较控制流：比较设计控制流图和实际程序生成的控制流图的差异。
- 选择、激活路径：在设计控制流图中选择某条路径，到实际的程序中激活这条路径，如果不能激活，则程序可能有错。
- 补充文档：根据以上检查项目，可以编制代码规则、规范和检查表等作为测试用例。
- 对照程序的规格说明，详细阅读源代码，比较实际的代码，从差异中发现程序的问题和错误。

代码检查法的检查主要是通过桌面检查、走查和代码审查的方式进行的。

1. 桌面检查

桌面检查是程序员对源程序代码进行分析、检验并补充相关的文档、发现程序中错误的过程。

2. 走查

走查是由程序员和测试员组成的审查小组通过逻辑运行程序发现问题的过程。小组成员要提前阅读设计规格书、程序文本等相关文档，利用测试用例使程序逻辑运行。

3. 代码审查

代码审查是由程序员和测试员组成的审查小组通过阅读、讨论、分析技术对程序进行静态分析的过程。

检查必须遵守规定代码的语法格式、语法规则，如排版、注释、标识符命名、可读性、变量、函数、过程、可测性、程序效率、质量保证、代码编辑、编译、审查、代码测试、维护、宏等各方面的编码要求。

3.3.2 静态结构分析法

静态结构分析法是测试者通过使用测试工具分析程序源代码的系统结构、数据结构、数据接口、内部控制逻辑等内部结构，生成函数调用关系图、模块控制流图、内部文件调用关系图等各种图形图表，清晰地标识整个软件的组成结构的方法。

静态结构分析法通过应用程序各函数之间的调用关系展示系统结构、列出所有函数，并用连线表示调用关系和作用。静态结构分析法主要分析以下内容：

- 可以检查函数的调用关系是否正确。
- 是否存在孤立的函数而没有被调用。
- 明确函数被调用的频繁度，对调用频繁的函数可以重点检查。

3.3.3 静态质量度量法

静态质量度量法是测试者通过软件质量、质量度量和度量规则进行分析的。

1. 软件质量

软件质量包括以下 6 个方面：

- 功能性（Functionality）。
- 可靠性（Reliability）。
- 可用性（Usability）。
- 有效性（Efficiency）。
- 可维护性（Maintainability）。
- 轻便性（Portability）。

2. 质量度量

将某一软件质量分为不同的分类标准，每个分类标准由一系列度量规则组成，每个规则分配一个权重，每个分类标准的取值由规则的取值与权重值计算得出，依据结果将软件质量分为 4 个等级：

- 优秀（Excellent）：符合模型框架中的所有规则（可以接受）。
- 良好（Good）：未大量偏离模型框架中的规则（可以接受）。
- 一般（Fair）：违背了模型框架中的大量规则（可以接受）。
- 较差（Poor）：无法保障正常的软件可维护性（不可以接受）。

3. 度量规则

度量规则使用代码行数、注释频度等参数度量软件的各种行为属性。

3.3.4 逻辑覆盖法

逻辑覆盖是通过对程序逻辑结构的遍历实现对程序的覆盖，主要涉及以下几个知识点：

- 测试覆盖率。
- 逻辑覆盖。
- 面向对象的覆盖。
- 测试覆盖准则。

1. 测试覆盖率

测试覆盖率是用于确定测试所执行到的覆盖项的百分比，其中的覆盖项是指作为测试基础的一个入口或属性，如语句、分支、条件等。

测试覆盖率可以表示出测试的充分性，在测试分析报告中可以作为量化指标的依据，测试覆盖率越高效果越好，但覆盖率不是目标，只是一种手段。

测试覆盖率包括功能点覆盖率和结构覆盖率：

- 功能点覆盖率主要用于表示软件已经实现的功能与软件需要实现的功能之间的比例关系。

- 结构覆盖率主要包括语句覆盖率、分支覆盖率、循环覆盖率、路径覆盖率等。

2. 逻辑覆盖

根据覆盖目标的不同和覆盖源程序语句的详尽程度，逻辑覆盖又可分为语句覆盖 SC（Statement Coverage）、判定覆盖 DC（Decision Coverage）、条件覆盖 CC（Condition Coverage）、条件判定组合覆盖 CDC（Condition/Decision Coverage）、多条件覆盖 MCC（Multiple Condition Coverage）、修正条件判定覆盖 MCDC（Multiple Condition Decision Coverage）、组合覆盖和路径覆盖。

（1）语句覆盖

语句覆盖是利用选择足够多的测试数据，使得程序中的每个可执行语句至少执行一次。

语句覆盖的缺点是：对程序执行逻辑的覆盖率低。

（2）判定覆盖

判定覆盖是通过设计足够多的测试用例，使得程序中的每一个判定至少获得一次“真”值和“假”值，或者使得程序中的每一个取“真”的分支或取“假”的分支至少经历一次，也称为“分支覆盖”。

判定覆盖的缺点是：主要对整个表达式最终取值进行度量，忽略了表达式内部取值。

（3）条件覆盖

条件覆盖是通过设计足够多的测试用例，使得程序中每个判定包含的每个条件的可能取值（真/假）都至少满足一次。

条件覆盖的缺点是：不能够满足判定覆盖。

（4）条件判定组合覆盖

条件判定组合覆盖是通过设计足够多的测试用例，使得程序中每个判定包含的每个条件的所有情况（真/假）至少出现一次，并且每个判定本身的判定结果（真/假）也至少出现一次。条件判定组合覆盖的测试用例一定同时满足判定覆盖和条件覆盖。

条件判定组合覆盖缺点是没有考虑单个判定对整体结果的影响，无法发现逻辑错误。

（5）多条件覆盖

多条件覆盖也称条件组合覆盖，是通过设计足够多的测试用例，使得每个判定中条件的各种可能组合都至少出现一次（以数轴形式划分区域，提取交集，建立最少的测试用例）。

多条件覆盖的缺点是：判定语句较多时，条件组合值比较多。

（6）修正条件判定覆盖

修正条件判定覆盖是对每一个程序模块的入口点和出口点都至少考虑被调用一次，从每个程序的判定到所有可能的结果值要至少转换一次。

（7）组合覆盖

组合覆盖是通过执行足够的测试用例，使得程序中每个判定的所有可能的条件取值组合都至少出现一次。满足组合覆盖的测试用例一定满足判定覆盖、条件覆盖和条件判定组合覆盖。

（8）路径覆盖

路径覆盖是利用设计足够多的测试用例，覆盖程序中所有可能的路径。

3. 面向对象的覆盖

面向对象的覆盖主要分为继承上下文覆盖和基于状态的上下文覆盖。

（1）继承上下文覆盖

由于传统的程序结构没有考虑面向对象的一些特性（如多态、继承和封装等），所以在面向对象领域，传统的程序结构覆盖必须被加强，以满足面向对象特性。继承上下文覆盖考虑在每个类的上下文内获得的覆盖率级别。

继承上下文的定义将基类上下文内例行程序的执行作为独立于继承类上下文内例行程序的执行。为了获得 100%继承上下文覆盖，代码必须在每个适当的上下文内被完全执行。

（2）基于状态的上下文覆盖

由于类的行为状态，每个类的行为在每个可能的状态中其性质是不同的。基于状态的上下文覆盖对应于被测类对象的潜在状态。

上下文覆盖把一个状态上下文内的一个例行程序达到 100%的基于状态的上下文覆盖，例行程序必须在每个适当的上下文（状态）内被执行。

4. 测试覆盖准则

测试覆盖准则主要讨论 ESTCA（Error Sensitive Test Cases Analysis）错误敏感测试用例分析和 LCSAJ（Linear Code Sequence and Jump）线性代码序列与跳转。

（1）错误敏感测试用例分析 ESTCA

ESTCA 覆盖准则在容易发生问题的地方设计测试用例，即重视程序中谓词（条件判断）的取值。这一规则虽然并不完备，但在普通程序中却是有效的。原因在于这是一种经验型的覆盖准则，规则本身针对了程序编写人员容易犯的错误，或是围绕着发生错误的频繁区域，从而提高了发现错误的命中率。规则具体如下。

- 规则 1：对于 A rel B（rel 可以是“<”、“=”和“>”）型的分支谓词（条件判断）的取值，应适当地选择 A 与 B 的值，使得测试执行到该分支语句时，A<B、A=B 和 A>B 的情况分别出现一次。
- 规则 2：对于 A rel C（rel 可以是“<”或是“>”，A 是变量，C 是常量）型的分支谓词，当 rel 为“<”时，应适当地选择 A 的值，使 A=C-M（M 是距 C 最小的机器容许正数，若 A 和 C 均为整数时，M=1）。同样，当 rel 为“>”时，应适当地选择 A，使 A=C+M。
- 规则 3：对外部输入变量赋值，使其在每一测试用例中均有不同的值与符号，并与同一组测试用例中其他变量的值与符号不一致。

（2）线性代码序列与跳转 LCSAJ

线性代码序列与跳转 LCSAJ 是指一组顺序执行的代码，以控制流跳转为结束点。可产生 4 层覆盖。

- 第一层：语句覆盖。
- 第二层：分支覆盖。
- 第三层：LCSAJ 覆盖。
- 第四层：两两层 LCSAJ 覆盖；一直到 N 个 LCSAJ。

LCSAJ 的起点是根据程序本身决定的。它的起点可以是程序第一行或转移语句的入口点，或是控制流可跳达的点。

在实施测试时，若要实现上述的层次 LCSAJ 覆盖，需要产生被测程序的所有 LCSAJ。

3.3.5 基本路径测试法

路径测试就是从一个程序的入口开始，执行所经历的各个语句的完整过程。从广义的角度讲，任何有关路径分析的测试都可以被称为路径测试。

完成路径测试的理想情况是做到路径覆盖，但对于复杂性强的程序要做到所有路径覆盖（测试所有可执行路径）是不可能的。

在不能做到所有路径覆盖的前提下，如果某一程序的每一个独立路径都被测试过，那么可以认为程序中的每个语句都已经检验过了，即达到了语句覆盖。这种测试方法就是通常所说的基本路径测试方法。

基本路径测试方法是在控制流图的基础上，通过分析控制结构的环路复杂度，导出执行路径的基本集，再从该基本集设计测试用例。基本路径测试方法包括以下 4 个步骤：

01 画出程序的控制流图。

02 计算程序的环路复杂度，导出程序基本路径集中的独立路径条数，这是确定程序中每个可执行语句至少执行一次所必须的测试用例数目的上界。

03 导出基本路径集，确定程序的独立路径。

04 根据 03 中的独立路径，设计测试用例的输入数据和预期输出。

下面重点介绍以下内容。

- 程序控制流图：描述程序控制流的一种图示方法。
- 程序环路复杂度：McCabe 复杂性度量。从程序的环路复杂性可导出程序基本路径集合中的独立路径条数，这是确定程序中每个可执行语句至少执行一次所必须的测试用例数目的上界。

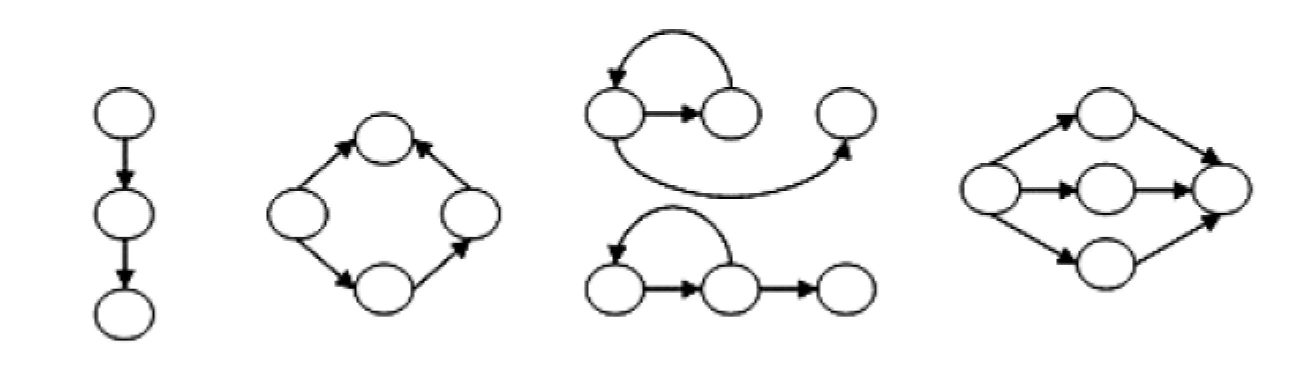
1. 程序控制流图

程序控制流图（可简称流图）是对程序流程图进行简化后得到的，它突出表示程序控制流的结构。程序控制流图是描述程序控制流的一种方式。控制流图的图形符号中，圆圈代表一个结点，表示一个或多个无分支的语句或源程序语句；程序控制流边和点圈定的部分叫做区域。当对区域计数时，图形外的一个部分也应记为一个区域；判断语句中的条件为复合条件时，即条件表达式由一个或多个逻辑运算符连接的逻辑表达式（a and b），则需要改变复合条件的判断为一系列只有单个条件的嵌套的判断。

节点由带标号的圆圈表示，可代表一个或多个语句、一个处理框序列和一个条件判定框（假

设不包含复合条件）。

控制流线由带箭头的弧或线表示，可称为边。它代表程序中的控制流。为了满足路径覆盖，必须首先确定具体的路径以及路径的个数。通常采用控制流图的边（弧）序列和节点序列表示某一条具体路径。程序控制流图如图 3-3 所示。



顺序结构　IF 选择结构　While 循环结构　CASE 多分支结构
Until 循环结构

图 3-3　程序控制流图

2. 程序环路复杂度

程序的环路复杂度也称为圈复杂度，它是一种为程序逻辑复杂度提供定量尺度的软件度量。

将环路复杂度用于基本路径方法，它可以提供：程序基本集的独立路径数量、确保所有语句至少执行一次的测试。独立路径是指程序中至少引入了一个新的处理语句集合或一个新条件的程序通路，包括一组以前没有处理的语句或条件的一条路径。通常环路复杂度以图论为基础，提供软件度量。可用如下方法来计算环路复杂度：

- 控制流图中区域的数量对应于环路复杂度。
- 给定控制流图 G 的环路复杂度 V(G)，定义为 V(G)=E-N+2，其中 E 是控制流图中边的数量；N 是控制流图中的节点数量。

计算环路复杂度的示意图如图 3-4 所示。

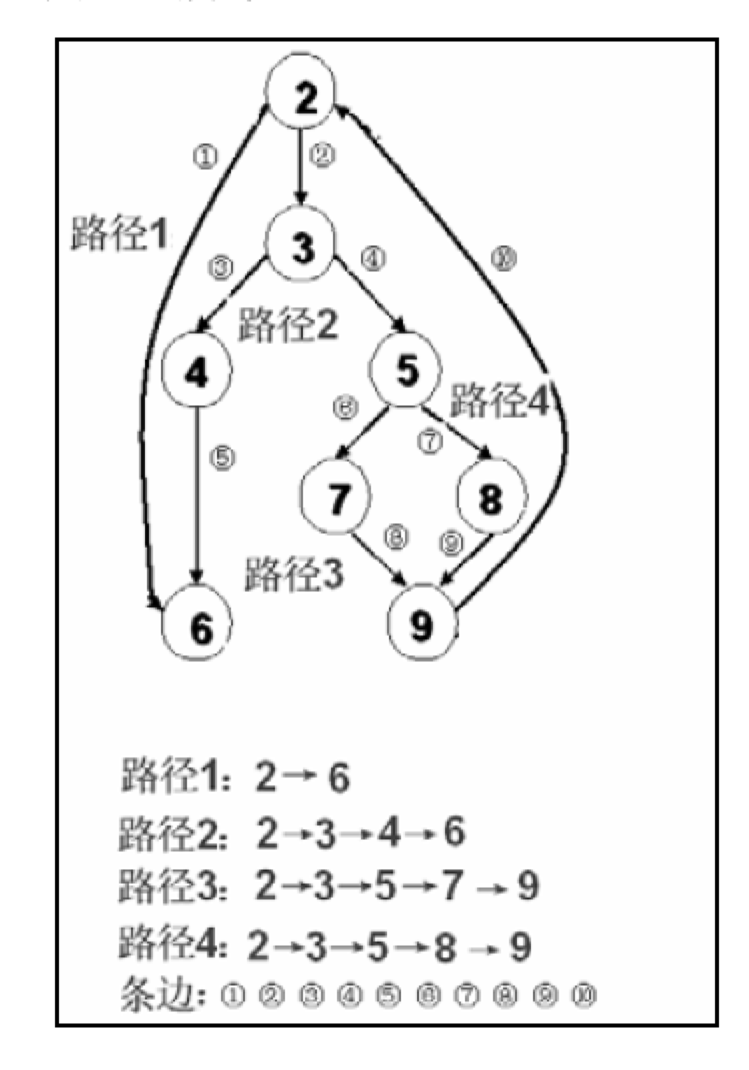


图 3-4　计算环路复杂度

图 3-4 中的节点数量 N=8，可以导出边的数量 E=10，用①、②、③、④、⑤、⑥、⑦、⑧、⑨、⑩编号表示。

V(G)=E-N+2=10-8+2= 10（边的数量）-8（节点的数量）+2=4

导出的独立路径用路径 1、路径 2、路径 3、路径 4 编号表示。

3.3.6 域测试法

域测试是一种基于程序结构的测试方法，基于对程序输入空间（域）的分析，选择测试点进行测试。主要错误包括域错误、计算型错误、丢失路径错误。

- 域错误：程序的控制流存在错误，对于某一特定的输入可能执行的是一条错误路径，这种错误称为路径错误，也叫做域错误。
- 计算型错误：对于特定输入执行的路径正确，但赋值语句的错误导致输出结果错误，称为计算型错误。
- 丢失路径错误：由于程序中的某处少了一个判定谓词而引起的丢失路径错误。

域测试法主要具有以下缺点：

- 为进行域测试对程序提出的限制过多。
- 当程序存在很多路径时，所需要的测试点很多。

3.3.7 符号测试法

符号测试法的基本思想是允许程序的输入不仅仅是具体的数值数据，而且包括符号值，符号值可以是基本的符号变量值，也可以是符号变量值的表达式。

符号测试法执行的是代数运算，可以作为普通测试的一个扩充，也可以看作是程序测试和程序验证的一个折衷办法。符号测试程序中仅有有限的几条执行路径。

符号测试法具有如下缺点：

- 分支问题不能控制。
- 二义性问题不能控制。
- 大程序问题不能控制。

3.3.8 Z 路径覆盖法

Z 路径覆盖法对循环机制进行了简化，减少了路径的数量，使得覆盖所有路径成为可能，简化循环意义下的路径覆盖称为 Z 路径覆盖，循环简化的目的是限制循环的次数，不考虑循环的形式和循环体实际执行的次数，简化后的循环测试只考虑执行循环体一次和零次（不执行）两种情况，即考虑执行时进入循环体一次和跳过循环体这两种情况。

3.3.9 程序变异测试法

程序变异是一种错误驱动测试。错误驱动测试是针对某类特定程序错误的，即专门测试某类错误是否存在，优点是便于测试对软件危害最大的可能错误，使测试效率具有较大的提高，并降低

了成本。

3.4　白盒测试的要求

本节主要从软件各层公用问题测试的要求、Java 语言测试的要求、数据类型测试的要求、SQL 语句测试的要求、界面测试的要求、数值对象测试的要求、业务对象测试的要求、数据管理对象测试的要求共 8 个方面进行介绍。

3.4.1　软件各层公用问题测试的要求

软件各层公用问题测试的要求主要包括如下 12 项。

1. 检查代码与设计对照表

检查代码与设计对照表时需要重点注意如下 4 点：

- 按软件需求检查 CRC 设计文档是否完全地实现了所有 CRC 中规定的内容，CRC 设计文档完备、没有错误。
- 按软件需求检查 UI 设计文档是否完全地实现了所有的 UI 设计的规定要求，UI 设计完备、没有错误。
- 按软件需求检查编码对照表设计文档是否完全地实现了软件所规定的内容，完备、没有错误。
- 检查代码名、代码位数、代码含义、姓名、编号、删除、追加、修改是否实现设计的规定要求。

2. 检查数据库

检查是否按软件需求创建了所需要的数据库，数据库的内容是否正确、完备。

3. 检查程序参数返回值

检查程序参数返回值时主要注意返回值的类型、个数、顺序是否正确。

4. 检查调用程序公用接口

检查程序公用接口是否有错误，数据类型、个数、顺序及返回值是否正确。

5. 检查子系统的设计

检查子系统的设计，重点注意如下 5 点：

- 主要功能。
- 主要流程。
- 输入内容。
- 输出内容。
- 交接口方式。

6. 检查数据库设计

检查数据库设计，重点注意如下 4 点：

- 数据库的主要内容。
- 数据库的逻辑划分。
- 数据库的安全措施。
- 数据库的更新备份与恢复方式。

7. 检查系统目标

检查系统目标时要重点注意问题和交付的结果。

（1）问题

问题方面要重点注意如下 8 点：

- 系统目标是否加以形式定义？还是它们是不严格地描述的，并且须经过解释或以后再定义？
- 新系统对该机构的基本操作是否会有重要影响？
- 新系统是否将代替现有系统？若是，那么当前的系统已经使用多久？在它之前还有多少其他系统？
- 是否指望新系统重新安排或删除任何工作职责？若是，这个问题的敏感程度如何？
- 是否要求一个临时的系统来满足即时的目的或者来消除与现有系统有关的不可容忍的问题？
- 对该项目可以分配什么资源？
- 所指望的新系统与技术发展水平密切到什么程度？
- 用户能够分配多少时间用于训练和开始工作？

（2）交付的结果

交付的结果方面要重点注意如下 6 点：

- 对系统目标的一个综合叙述。
- 对所要求项目工作的一般范围和等级的叙述，包括初步价格和资源估计。
- 从改变、消除或替代几方面考虑的对当前系统的叙述。
- 对所指望的项目阶段划分和对项目的总体协调方法的描述。
- 对由该系统引起的预料的机构改变的程度和影响的初步陈述。
- 对在这个要求的系统中每个参加的用户部门和主要用户组的作用及职责的注释。

8. 检查数据元素的结构

检查数据元素的结构要重点注意问题和交付的结果。

（1）问题

问题方面要重点注意如下 12 点：

- 当前的数据元素、文卷、表格、过程等是否完全地编制了文件资料？
- 当前的数据元素和结构是否是合理的、一致的和可应用的？
- 数据库清晰到什么程度？
- 用户是否有一张他们希望在新系统中见到的新数据元素表？增加这些数据元素是否可行？
- 在现有系统的数据库和该机构中其他应用的数据库之间有多大的冗余？对于数据库的任何元素，任何其他应用是否是一个更合理的存储？
- 为了适应新系统需要，当前的数据结构是否具有足够的灵活性？
- 把当前数据库转换成一个新的数据库会有多少困难？为了完成一个没有问题的转换将需要多少出错测试？
- 在现有数据库中通常要进行多少维护？
- 来自这个数据库的大批数据文件是否能够或应该转换？
- 当前数据库中有多少是实际使用的？谁在使用？
- 在数据文卷方面曾遇到些什么重要的故障和错误？对它们是怎样处理的？
- 该数据库已修改多少次？采用了什么方法？

（2）交付的结果

交付的结果方面要重点注意如下 3 点：

- 所有数据元素、文卷和支撑性的数据结构的一组综合性的格式和内容定义。
- 对当前数据库内容的一个评价，重点是清洁度、差错、无用区域、冗余度、转换以及将来的使用价值。
- 对新系统所预料的数据元素和结构进行所期望的改变、增加、删除和其他修改的表。

9. 检查用户访问

检查用户访问需要重点注意问题和交付的结果。

（1）问题

问题方面要重点注意如下 12 点：

- 所有用户是否都已标识？
- 对所涉及的每一个用户层，是否都有一个正式的访问计划？
- 为每个用户层的访问是否制定了问题和目的清单？
- 上级管理部门是否支持和宣扬这种访问？上级管理部门对被访问者们的合作关系是否作了强有力的布置？
- 所有访问的日程安排是否都在可接受的时间范围内？
- 访问者在有效的访问技术方面是否受过训练？
- 所有预定的访问是否都已完成？被取消、被中断的或者被忘记的访问是否已经重新安排并进行？
- 访问者对每次访问是否作了充分的笔记并且写了评价？
- 访问者是否参考了笔记、印象以及其他观察？这些细节是否编写了文卷资料？

- 对被访问者是否给了足够的反馈信息？例如小结报告、笔记等。
- 在初始访问期间，当暴露出特殊的问题或条件时，继续追究性的访问是否进行？
- 是否随时向管理部门报告了有关访问进程、任何暴露的问题，以及不配合的用户的情况？

（2）交付的结果

交付的结果方面要重点注意如下 6 点：

- 访问结果的文件资料。
- 一份访问小结报告，包括一致的答案和重要分歧两个方面。
- 用户对于该系统的态度和地位的内部分析。
- 关于访问所得结论和参加者的协作关系的管理报告。
- 随着问题、重点和其他访问方针改变检验访问的结果。
- 关于任何未完成访问的说明。

10. 检查结构上的分析

检查结构上的分析需要重点注意问题和交付的结果。

（1）问题

问题方面要重点注意如下 12 点：

- 对于所选用方案是否定义了所有数据元素、数据流，以及所要求的处理步骤？
- 是否定义和评价了新系统将造成的过程和机构的变化？
- 输入文件和输出文件的内容及用途是否已按一般方法定义？
- 关于新系统的设备要求是否估计？
- 是否存在一张期望的系统模块表？
- 是否有一个初步的数据转换计划？
- 是否有一张正在生成的整个系统流程图？
- 有关的事务性过程是否已提出要点？
- 估算的数据量是多少？
- 是否正在考虑数据的安全性和精确度要求？
- 关于新方法的测试过程是否已完全确定？
- 是否已有一份初步的系统实现计划？

（2）交付的结果

交付的结果方面要重点注意如下 7 点：

- 一份关于所建议的系统方法的报告。
- 一份系统流程图。
- 一份用户操作和职责的流程图。
- 关于分析结论的详细报告。
- 价格利益的分析报告。
- 初步的测试计划。

- 初步的实现计划。

11. 检查程序是否冗余

检查程序是否冗余时，要重点注意对于程序中的大量重复内容，是否使用了专门的类来实现。

12. 检查代码整体规范

检查代码整体规范时，要重点注意代码是否自始至终遵循《程序员开发手册》和《编码规范》中要求的格式、调用约定、结构等。

3.4.2 Java 语言测试的要求

Java 语言测试检查时主要有如下 9 点注意事项：

- 检查 Java 语言的下标是否有下标变量越界错误。
- 检查 Java 语言的除数是否包含有除零（n/0）错误的可能。
- 检查字符串。
- 检查字符串连接符“+”。
- 检查浮点值、整型值的应用是否有错误。
- 检查 switch 语句的应用是否有错误。
- 检查 if 语句的应用是否有错误。
- 检查循环语句的应用是否有错误。
- 检查数值范围是否存在溢出错误。

3.4.3 数据类型测试的要求

数据类型测试检查时主要有如下 5 点注意事项：

- Null 转换。在设置值对象 VO 时，在 VO 内部是否将空串转换为 Null，数值型数据（整数、浮点数）Null 转换为 0。
- 检查控件数据类型的转换，即查看控件数据类型是否与对应字段数据类型一致。
- 检查精度型（8 位）、双精度型（20 位）控件的范围控制。
- 检查小数位数的设置。
- 检查禁止输入字符的设置。

3.4.4 SQL 语句测试的要求

SQL 语句测试检查时主要有如下 21 点注意事项。

1. 检查数据库文件集

SQL 系统数据库内在的每个数据库都有自己的文件集，而且不与其他数据库共享这些文件，对数据库文件集的说明如表 3-1 所示。

表 3-1　SQL 系统数据库内在的数据库文件集

数据库文件	物理文件名	默认大小，典型安装
master 主数据	Master.mdf	110MB
master 日志	Mastlog.ldf	125MB
tempdb 主数据	Tempdb.mdf	80MB
tempdb 日志	Templog.ldf	5MB
model 主数据	Model.mdf	75MB
model 日志	Modellog.ldf	75MB
msdb 主数据	Msdbdata.mdf	120MB
msdb 日志	Msdblog.ldf	225MB

SQL Server 2000 中的每个数据库都包含系统表，用来记录 SQL Server 组件所需的数据。SQL Server 的操作能否成功取决于系统表信息的完整性，因此 Microsoft 不支持用户直接更新系统表中的信息。

2. 检查 SQL 数据库对象

需要着重进行强调和解释的有以下几项：

- 每个数据库对象都有拥有者。
- Table：是 DB 的基本单位，由行和列组成，用于存储数据。
- Data Type：限制输入到表中的数据类型。
- Constraint：有主键、外键、惟一键、缺省和检查 5 种。
- Default：自动插入常量值。
- Rule：限制表中列的取值范围。
- Trigger：一种特殊类型的存储过程，当有操作影响到它保护的数据时，自动触发执行。
- Index：提高查询速度。
- View：查看一个或多个表的一种方式。
- Stored Procedure：一组预编译的 SQL 语句，可以完成指定的操作。

3. 检查 SQL 语句的书写规范

检查 SQL 语句的书写规范时主要有如下 3 点注意事项：

- 禁止使用“select * from”语法。
- 禁止使用“insert into table_name values(?,?,…)”语法。
- 统一使用“insert into table_name (col1,col2,…) values(?,?,…)”语法。

4. 检查 SQL 语句的类型转换

检查 SQL 语句的类型转换时，主要避免显式或隐含的类型转换。

5. 检查容量规划

检查容量规划时主要有如下 5 点注意事项：

- 估计数据库的尺寸。

- 估计表中的数据量。
- 计算每行中的字节数。
- 计算行中的总字节数。
- 确定数据页中的行数。

6. 检查变量定义

检查变量定义时，注意检查局部变量和全局变量的定义。

7. 检查 if 语句的定义

如果 if 或者 else 控制的不是一行，就要通过“begin…end”来确定边界。

8. 检查数据类型

每个列、局部变量、表达式和参数都有一个相关的数据类型，这是指定对象可持有的数据类型（整型、字符、money 等）的特性。尽量不使用定长的数据类型。

9. 检查数据完整性

检查数据完整性时主要检查如下 4 项。

- 实体完整性：主键字段不能为空值。
- 参照完整性：外键字段必须是另一张表主键的有效值或空值。
- 域完整性。
- 自定义完整性。

10. 检查临时表

局部临时表仅仅限于建立它的人可以使用。全局临时表则没有此限制，其他用户也可以使用。创建它的人断开后，其他人无法打开，但已经在用的可继续使用。

11. 检查集合的合并

检查集合的合并时，要着重检查集合合并的内容是否正确、完备。

12. 检查隔离等级

检查隔离等级时，需要注意以下 3 点：

- 保证不会读到别人修改的数据。
- 保证已读取的数据不可更改。
- 保证使用到的数据表不被更改。

13. 检查安全性管理

检查安全性管理时，需要注意以下两点：

- 检查两种认证机制（NT 认证机制+SQL Server 认证机制）。
- 检查固定服务器角色和固定数据库角色。

其中，对固定服务器角色的说明如表 3-2 所示。

表 3-2　固定服务器角色

固定服务器角色	描述
sysadmin	在 SQL Server 中可进行任何活动。该角色的权限可跨越所有其他固定服务器角色
serveradmin	配置服务器范围
setupadmin	添加和删除链接服务器，并执行某些系统存储过程
securityadmin	管理服务器登录
processadmin	管理在 SQL Server 实例中运行的进程
dbcreator	创建和改变数据库
diskadmin	管理磁盘文件
bulkadmin	执行 BULK INSERT 语句

对固定数据库角色的说明如表 3-3 所示。

表 3-3　固定数据库角色说明

固定数据库角色	描述
db_owner	进行所有数据库角色的活动，以及数据库中的其他维护和配置活动。该角色的权限可跨越所有其他固定数据库角色
db_accessadmin	在数据库中添加或删除 Windows NT 4.0/Windows 2000 组和用户以及 SQL Server 用户
db_datareader	查看来自数据库中所有用户表的全部数据
db_datawriter	添加、更改或删除来自数据库中所有用户表的数据
db_ddladmin	添加、修改或删除数据库中的对象（运行所有 DDL）
db_securityadmin	管理 SQL Server 2000 数据库的角色和成员，并拥有管理数据库中的语句和对象权限
db_backupoperator	具有备份数据库的权限
db_denydatareader	具有拒绝选择数据库数据的权限
db_denydatawriter	具有拒绝更改数据库数据的权限

14. 检查调度作业

可在以下 5 种情况下进行调度作业检查：

- 每当 SQL Server 代理程序启动时。
- 每当计算机的 CPU 使用率处于定义为空闲状态的水平时。
- 在特定日期和时间运行一次。
- 按循环调度运行。
- 响应警报。

15. 检查指定作业响应

可查看以下 3 种响应是否正常：

- 使用电子邮件、电子呼叫或 net send 消息通知操作员。
- 将事件消息写入 Microsoft Windows 应用程序日志。
- 自动删除作业。若确信不需要再次运行该作业，可以使用这种响应。

16. 检查报警管理

可通过以下操作检查报警管理是否正常：

- 通知一个或多个操作员。
- 将事件转发给其他服务器。
- 执行作业。

17. 检查备份

检查以下功能是否完备：

- 允许动态备份。
- 执行和存储备份。

18. 检查数据库还原

可通过以下操作检查数据库还原功能是否正常：

- 恢复进程。
- 对备份的内容进行恢复。
- 从不同的备份类型中还原数据库。
- 恢复损坏的系统数据库。

19. 检查数据传输

可以通过以下操作检查数据传输是否正常：

- 导入/导出数据，包括移动、复制、归档和迁移数据。
- 更改数据格式。
- 转换和映射数据。
- 验证数据的有效性。
- 调度操作。
- 在异构环境之间导入/导出数据。

20. 检查分布式数据

检查分布式数据时，需要注意以下 4 点：

- 分布数据复制。
- 复制代理（快照代理、分发代理、日志读取器代理、合并代理程序、队列读代理程序）。
- 事务复制。
- 合并复制。

21. 检查函数

检查函数时，需要查看是否有动态创建函数的情况，这是不允许的。

3.4.5 界面测试的要求

界面 UI 检查主要包括如下 16 项内容。

1. 检查继承类

检查继承类时，需要主要查看每个界面类都实现软件所规定的内容，检查是否完备、正确。

2. 检查按钮

检查按钮时，需要查看以下几项：

- 检查界面类添加的按钮。
- 检查添加按钮的属性。
- 检查添加按钮组的属性。
- 检查响应按钮。

3. 检查界面是否规范

检查界面规范时，需要注意以下几项：

- 菜单、控件的一致性测试主要查看菜单与控件是否齐全、控件的类型是否正确。
- 界面的整体布局测试主要查看是否协调，颜色、尺寸是否合理。具有操作逻辑的控件，其摆放的先后位置是否合理。
- 图标、标题、标签测试主要查看各种图标使用是否符合规范，标题文本、标签文本是否正确合理。
- 控件的鼠标提示文本测试主要查看是否有鼠标提示，提示文本是否正确合理（针对重要的控件）。
- 最小化、最大化、关闭按钮是否有效。
- 菜单各种功能按钮是否有效。
- 单击“增加”按钮是否刷新界面，处于待输入状态时，“增加”按钮、“删除”按钮是否变灰，并激活“取消”、“确定”按钮。光标是否定位于第一个文本域内。
- 文本域内是否能输入正常长度、正确数据类型的数据（各提示键显示的内容是否正确）。
- 输入非正常的数据，系统是否有相应的错误提示，术语是否正确。
- 在增加状态下单击“取消”按钮，是否放弃当前操作，返回前一界面，并激活“增加”按钮。
- 单击“删除”按钮是否能删除选中的记录。
- 状态栏显示是否正确。
- 是否响应键盘事件（Enter 键、Delete 键、Tab 键）。
- 单击“退出”按钮是否能正常终止主应用程序。

4. 检查界面的初始化状态

检查界面的初始化状态时，可着重检查以下 3 项内容：

- “增加”、“退出”、“浏览”按钮是否被激活，“取消”按钮是否变灰。

- 界面是否自动定位于最后（或最前）一条记录。
- 各种标签文本和图标提示文本是否正确。

5. 检查编辑控件（除功能按钮以外的控件）

检查编辑控件时，可着重检查以下 8 项内容：

- 显示控件和编辑控件应该加以区分。
- 属性是否齐全。
- 控件的应用是否合理。
- 能否正常接收数据，对非法类型的数据是否进行了控制。
- 允许输入的数据长度是否符合要求。
- 控件的边界状态是否设定，如文本框的滚动条等。
- 是否有快捷键，快捷键是否有效。
- 主要的功能按钮是否响应键盘事件。

6. 检查通用对话框

检查通用对话框时，可着重检查以下 3 项内容：

- 图标、标题是否正确。
- 标签、提示文本是否正确合理。
- 功能按钮是否齐全、合理有效。

7. 检查状态栏

检查状态栏时，可着重检查以下 3 项内容：

- 增加、修改、保存、删除等操作是否能在状态栏显示其操作状态（成功、失败等）。
- 随着操作的不同，状态栏是否能进行相应的变化。
- 在查询大数据量时，是否有提示用户等待窗口。

8. 检查业务功能

检查业务功能时，可着重检查以下 4 项内容：

- 测试各菜单和功能按扭的缺省状态（变灰与激活）是否合理。
- 各种控件的缺省值是否正确。
- 对于母子表的界面，注意母子表是否能同步显示，显示的明细记录是否正确。
- 新建、保存、删除、查询、浏览、退出是否正确。

9. 检查界面校验

检查界面校验时，可着重检查以下 13 项内容：

- 各功能按钮和菜单状态变化是否正确。
- 界面的编辑框是否刷新（注意合理的保留值不应刷新）。
- 光标定位是否合理。
- 能否输入合法的数据。

- 能否正常地调出参照框，并导入所需要的数据，包括下拉框、参照对话框、右键菜单等。
- 能否正常修改或清除数据。
- 在没保存所编辑的记录时进行其他操作，系统是否提示保存新增记录，对话框文本是否正确合理。
- 单击“保存”按钮后，是否进行全面的逻辑校验（与设计文档相符），与正常的业务逻辑是否能保持一致。
- 提示文本是否正确合理，对话框能否正常操作。
- 退出对话框后光标定位是否合理。
- 能否正常保存数据。
- 界面数据显示是否正确。
- 菜单、其他功能按钮及控件状态变化是否合理。

3.4.6　数值对象测试的要求

数值对象（Valuble Object）测试时主要注意如下内容：

- 检查数字转换为其字符串表示形式的数值格式化方法，是否完全地实现了软件所规定的内容，没有错误。
- 检查以参数的形式传递对象的接口，是否完全地实现了软件所规定的内容，没有错误。
- 检查参数以合法的方式提供格式化服务，是否完全实现了软件所规定的内容，没有错误。
- 检查数据库自动生成数值序列功能，是否完全实现了软件所规定的内容，没有错误。
- 检查布尔对象转换值、Null、未定义、0或false均转换成布尔对象的方法，是否完全实现了软件所规定的内容，没有错误。
- 检查数值函数对象是否完全实现了软件所规定的内容，没有错误。
- 检查将非字母、数字字符转换成ASCII码编码函数，是否完全实现了软件所规定的内容，没有错误。
- 检查将ASCII码转换成字母、数字字符译码函数，是否完全实现了软件所规定的内容，没有错误。
- 检查不同进制（二、八、十六）的数值转换成十进制整数转换函数，是否完全实现了软件所规定的内容，没有错误。
- 检查将数值字符串转换成浮点数的转换函数，是否完全实现了软件所规定的内容，没有错误。
- 检查“/”除号、“<”小于、“<=”小于等于、“<>”不等于、“=”等于、“>”大于、“>=”大于等于、and逻辑与、not逻辑非、or逻辑或，是否完全实现了软件所规定的内容，没有错误。

3.4.7　业务对象测试的要求

业务对象（Business Object）测试的要求主要有如下内容：

- 检查处理应用程序的业务逻辑和业务校验，是否完全实现了软件所规定的内容，完备、没有错误。
- 检查允许与其他层相互作用的接口，是否完全实现了软件所规定的内容，完备、没有错误。

- 检查管理业务层级别的对象的依赖，是否完全实现了软件所规定的内容，完备、没有错误。
- 检查函数之间（考虑复用）的功能是否独立、效率如何、功能是否完整，且从全局角度来看是否雷同。
- 检查类中是否存在名称相同且参数个数相同的方法（类中不能有名称相同且参数个数相同的两个方法同时存在）。
- 检查抛出异常功能(BO 的所有业务方法都必须抛出异常，否则将不能生成 EJB 辅助代码)。
- 检查 BO 对象中的使用环境变量时，使用方法是否有误。
- 检查工具生成代码是否可用。
- 检查 BO 类中方法的命名是否反映该方法的业务含义。
- 检查 BO 类是否生成供客户端调用 BS 端的代码。
- 检查向数据库插入一条记录时，是否为它提供惟一主键（OID）。

3.4.8　数据管理对象测试的要求

数据管理对象测试的要求主要有如下内容。

- 检查数据库的利用效率。
- 检查 DMO 类中方法的完整性（DMO 类中应包含 insert()、delete()、update()方法，还可以包括其他的查询方法）。
- 检查数据库连接，是否完全实现了设计规定的要求，没有错误。
- 检查数据库资源的获得和释放是否正常。
- 检查是否可以尽量使用 VO 数组，如果 DMO 类中的方法需要返回业务数据，则通常是 VO 对象或 VO 对象的数组（或集合）。当客户端需要多个 VO 对象时，是否尽量使用 VO 数组的形式返回，以提高数据库和网络效率，不要将多个 VO 一个一个地查询和返回。
- 检查自动生成代码的调整。
- 检查条件拼写语句，没有错误。

3.5　白盒测试的工具

白盒测试工具一般根据测试工具原理的不同，可分为代码测试工具、静态测试工具和动态测试工具。

3.5.1　代码测试工具

1. 代码测试概述

代码测试（Code Test）工具是一个硬件辅助软件的测试与分析工具，为追踪嵌入式应用程序、分析软件性能、测试软件的覆盖率以及存储体的动态分配等提供了一个实时在线的高效率、可共享的网络工具。测试中发现的缺陷可以定位到代码级，可同时监视整个应用程序，这就避免了在选择程序的哪部分来观测以及如何配置相应工具来对各部分进行测试时带来的困难。即便是在程序超出高速缓存（Cache）或被动态再分配时，仍能生成可靠的追踪及测试结果。Code Test 能够同时测试出软件的性能、代码覆盖、存储器动态分配、捕获函数的每一次运行，无论是在检测一个局部的软

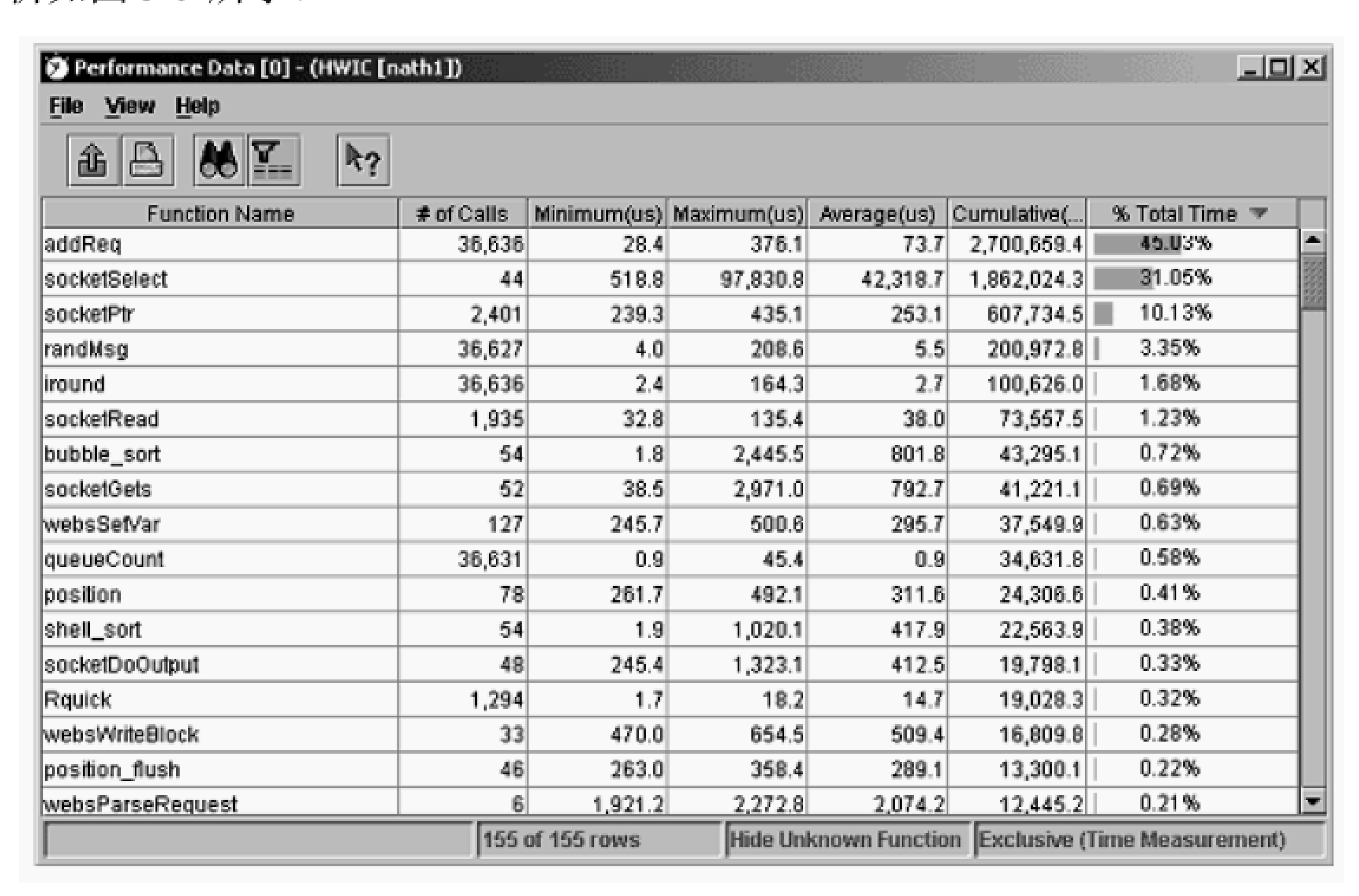
件模块还是整个软件系统测试，都能给整个开发和测试团队带来高品质的测试手段。

2. 代码测试工具的 4 大功能

（1）性能分析

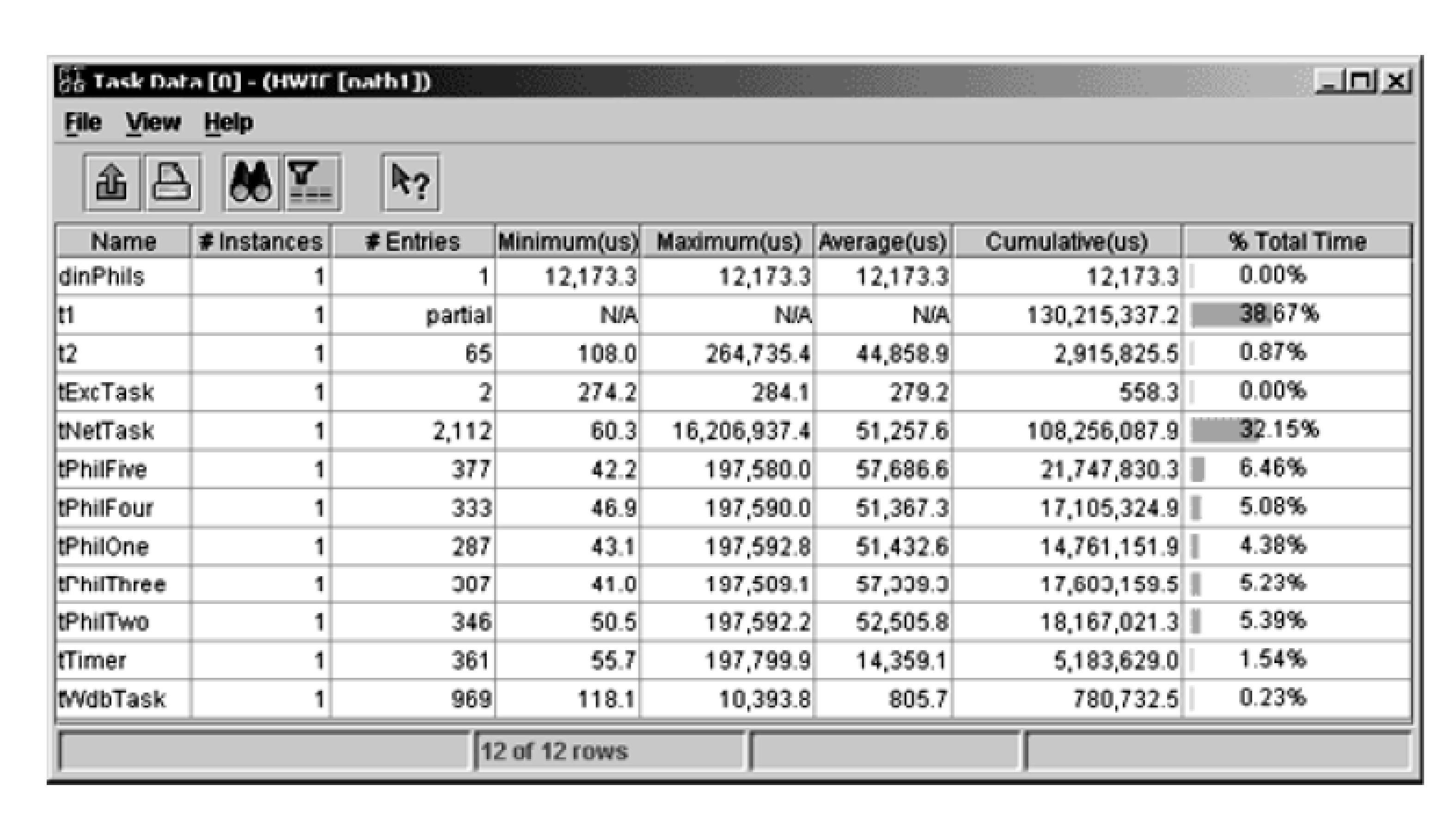
Code Test 能同时对 128000 个函数和 1000 个任务进行性能分析，精确计算出每个函数或任务执行的最大时间、最小时间和平均时间，精确度可以达到 50ns；能够精确地显示各函数或任务之间的调用情况，为嵌入式应用程序的优化提供依据，使软件工程师可以有针对性地优化某些关键性的函数或模块，帮助发现系统瓶颈、优化系统和提升系统性能。

性能分析如图 3-5 所示。



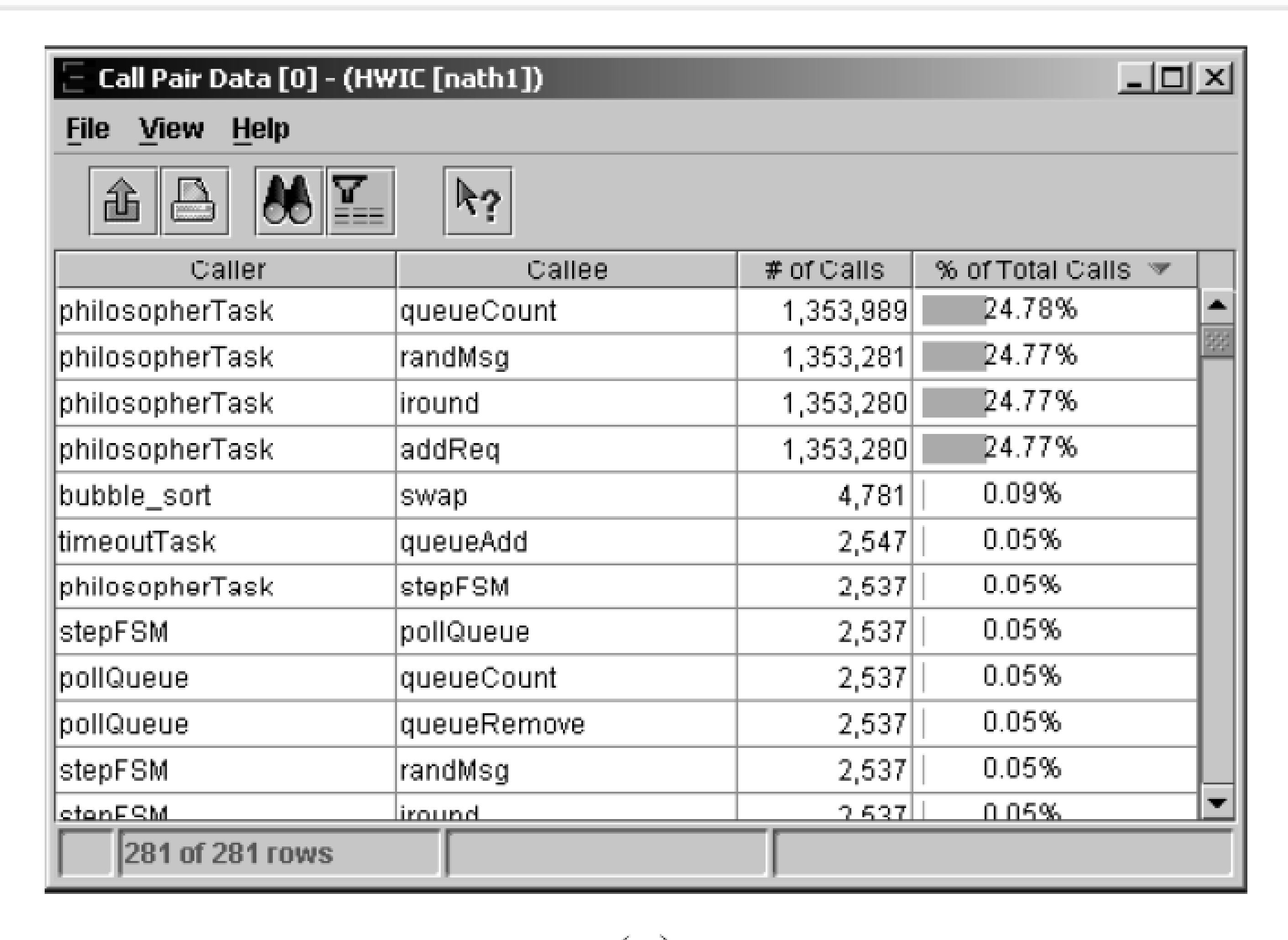
Function Name	# of Calls	Minimum(us)	Maximum(us)	Average(us)	Cumulative(...	% Total Time
addReq	36,636	28.4	376.1	73.7	2,700,659.4	45.03%
socketSelect	44	518.8	97,830.8	42,318.7	1,862,024.3	31.05%
socketPtr	2,401	239.3	435.1	253.1	607,734.5	10.13%
randMsg	36,627	4.0	208.6	5.5	200,972.8	3.35%
iround	36,636	2.4	164.3	2.7	100,626.0	1.68%
socketRead	1,935	32.8	135.4	38.0	73,557.5	1.23%
bubble_sort	54	1.8	2,445.5	801.8	43,295.1	0.72%
socketGets	52	38.5	2,971.0	792.7	41,221.1	0.69%
websSetVar	127	245.7	500.6	295.7	37,549.9	0.63%
queueCount	36,631	0.9	45.4	0.9	34,631.8	0.58%
position	78	261.7	492.1	311.6	24,306.6	0.41%
shell_sort	54	1.9	1,020.1	417.9	22,563.9	0.38%
socketDoOutput	48	245.4	1,323.1	412.5	19,798.1	0.33%
Rquick	1,294	1.7	18.2	14.7	19,028.3	0.32%
websWriteBlock	33	470.0	654.5	509.4	16,809.8	0.28%
position_flush	46	263.0	358.4	289.1	13,300.1	0.22%
websParseRequest	6	1,921.2	2,272.8	2,074.2	12,445.2	0.21%

（a）



Name	# Instances	# Entries	Minimum(us)	Maximum(us)	Average(us)	Cumulative(us)	% Total Time
dinPhils	1	1	12,173.3	12,173.3	12,173.3	12,173.3	0.00%
t1	1	partial	N/A	N/A	N/A	130,215,337.2	38.67%
t2	1	65	108.0	264,735.4	44,858.9	2,915,825.5	0.87%
tExcTask	1	2	274.2	284.1	279.2	558.3	0.00%
tNetTask	1	2,112	60.3	16,206,937.4	51,257.6	108,256,087.9	32.15%
tPhilFive	1	377	42.2	197,580.0	57,686.6	21,747,830.3	6.46%
tPhilFour	1	333	46.9	197,590.0	51,367.3	17,105,324.9	5.08%
tPhilOne	1	287	43.1	197,592.8	51,432.6	14,761,151.9	4.38%
tPhilThree	1	307	41.0	197,509.1	57,339.3	17,603,159.5	5.23%
tPhilTwo	1	346	50.5	197,592.2	52,505.8	18,167,021.3	5.39%
tTimer	1	361	55.7	197,799.9	14,359.1	5,183,629.0	1.54%
tWdbTask	1	969	118.1	10,393.8	805.7	780,732.5	0.23%

（b）



Caller	Callee	# of Calls	% of Total Calls
philosopherTask	queueCount	1,353,989	24.78%
philosopherTask	randMsg	1,353,281	24.77%
philosopherTask	iround	1,353,280	24.77%
philosopherTask	addReq	1,353,280	24.77%
bubble_sort	swap	4,781	0.09%
timeoutTask	queueAdd	2,547	0.05%
philosopherTask	stepFSM	2,537	0.05%
stepFSM	pollQueue	2,537	0.05%
pollQueue	queueCount	2,537	0.05%
pollQueue	queueRemove	2,537	0.05%
stepFSM	randMsg	2,537	0.05%
stepFSM	iround	2,537	0.05%

（c）

图 3-5　性能分析图

（2）测试覆盖率分析

Code Test 能够提供程序的总体概况，在系统真实的环境下，可以从单元级、集成级、系统级以及产品终端现场阶段进行嵌入式软件的分析与测试。由于 Code Test 是一种完全地交互式工具，测试者可以在实时的系统环境下进行简单语句覆盖 SC（Statement Coverage）、决策覆盖 DC（Decision Coverage）和条件决策覆盖 MC/DC（Modified Condition/Decision Coverage）的代码覆盖率测试，帮助测试工程师掌握当前的代码测试覆盖情况、指导测试用例的编写、加速测试进程和产品风险的评估过程。

- SC（Statement Coverage）：每执行一条源代码语句即视为这一条语句 100%覆盖。
- DC（Decision Coverage）：对于每一条条件语句（if/switch/do/while/for）的每一个不同的决策值至少出现一次才可被视为 100%覆盖。
- MC/DC（Modified Condition/Decision Coverage）：对于每一条条件语句的条件，在分解为乘积最小项的加法的真值表中，至少每一个最小项的条件都被满足一次才可视为完全覆盖。

Code Test 覆盖率信息包括程序实际执行的所有内容，而不是采样的结果，它以不同的颜色来区分运行和未运行的代码，Code Test 可以跟踪超过一百万个分支点，特别适用于测试大型嵌入式软件。

Code Test 还能够生成一个融合多种测试结果的综合性报告，以使测试者看到整套测试的总体效果。

测试覆盖率的示意图如图 3-6 所示。

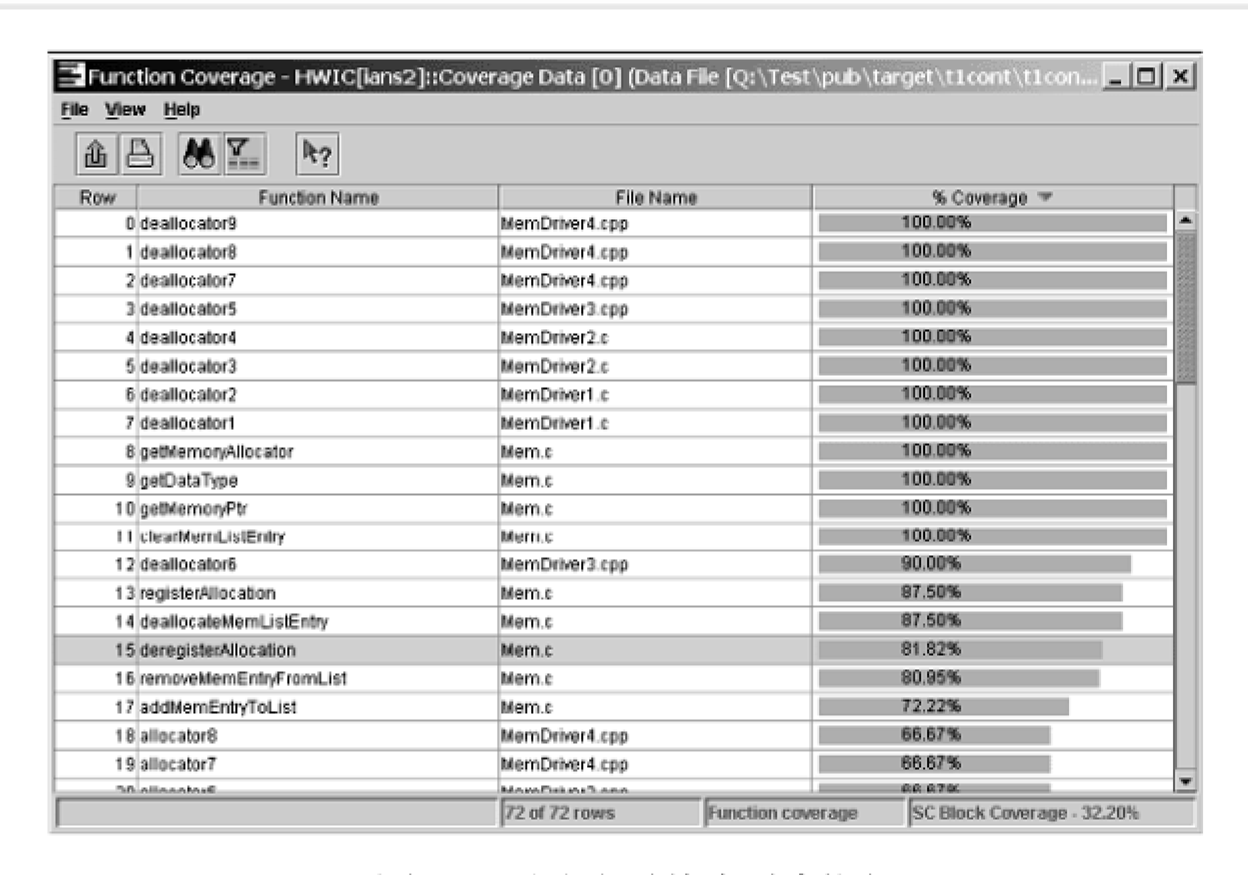


图 3-6　测试覆盖率分析图

（3）内存分析

在 Code Test 诞生之前，存储器分配情况是难以追踪观测的。Code Test 可以动态追踪内存分配，报告内存出错和相应的原始数据，能够显示有多少字节的存储器被分配给了程序的哪一个函数。这样就不难发现哪些函数占用了较多的存储空间、哪些函数没有释放相应的存储空间等，不仅可以在程序运行时报告每条语句分配多少字节的内存，而且可以鉴别 20 多种内存分配的错误。测试者还可以观察到存储体分配情况随着程序运行动态地增加和减少，即 Code Test 可以统计出所有的内存分配情况。随着程序地运行，Code Test 能够指出存储体分配的错误，测试者可以同时看到其对应的源程序内容，如 Code Test 能够指出 20 多种内存分配错误，并报告发生错误的函数和代码行。

动态内存分配的示意图如图 3-7 所示。

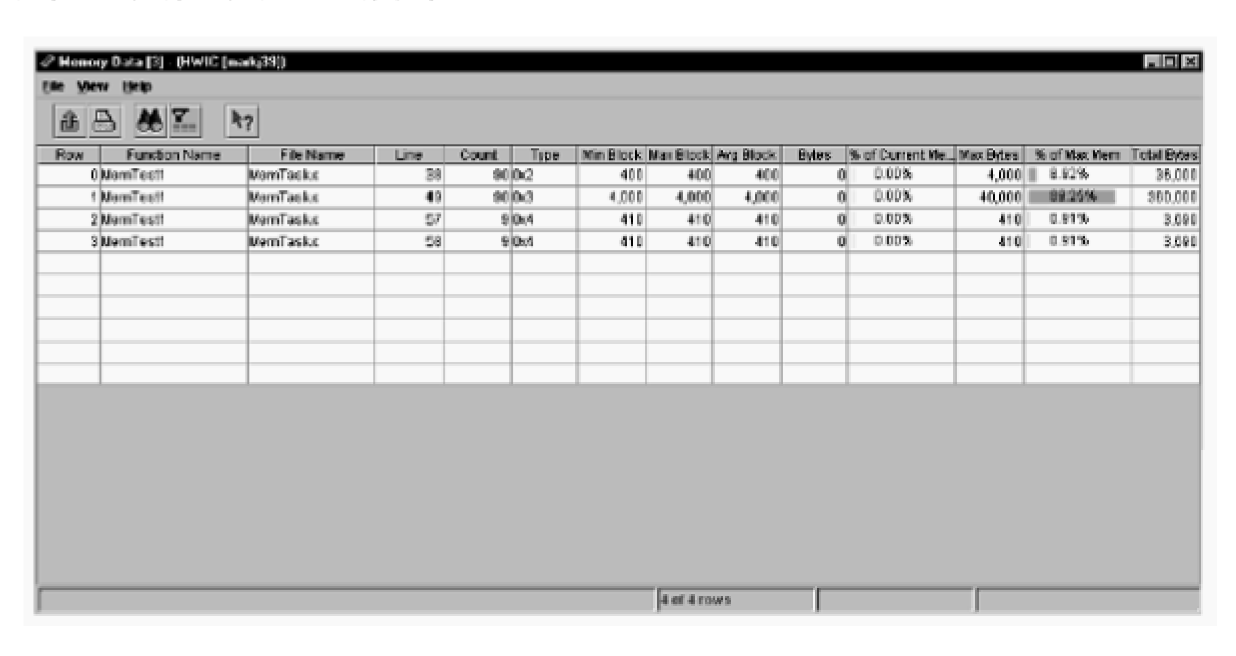


图 3-7　动态内存分配分析图

（4）执行追踪分析

Code Test 可以按源程序、控制流和高级模式来追踪嵌入式软件。

- 按源程序追踪嵌入式软件：Code Test 可以提供 400K 的追踪缓冲空间，最大追踪深度可达 150 万条源程序。
- 按控制流追踪嵌入式软件：增加了可执行函数中每一条分支语句的显示。
- 按高级模式追踪嵌入式软件：显示的是 RTOS 的事件和函数的进入/退出，为测试者提供一个程序流程的大框图；源级追踪则又增加了对被执行的全部语句的显示。

在以上三种模式下，均会显示详细的内存分配情况，包括在哪个代码文件的哪一行、哪个函数调用了内存的分配或释放函数、被分配的内存的大小和指针、被释放的内存指针、出现的内存错误等。可以显示运行过程中程序的实际情况、帮助查找程序的错误所在，使软件工程师有针对性地优化某些关键性的函数或模块，并改善整个软件的总体性能。

执行追踪的示意图如图 3-8 所示。

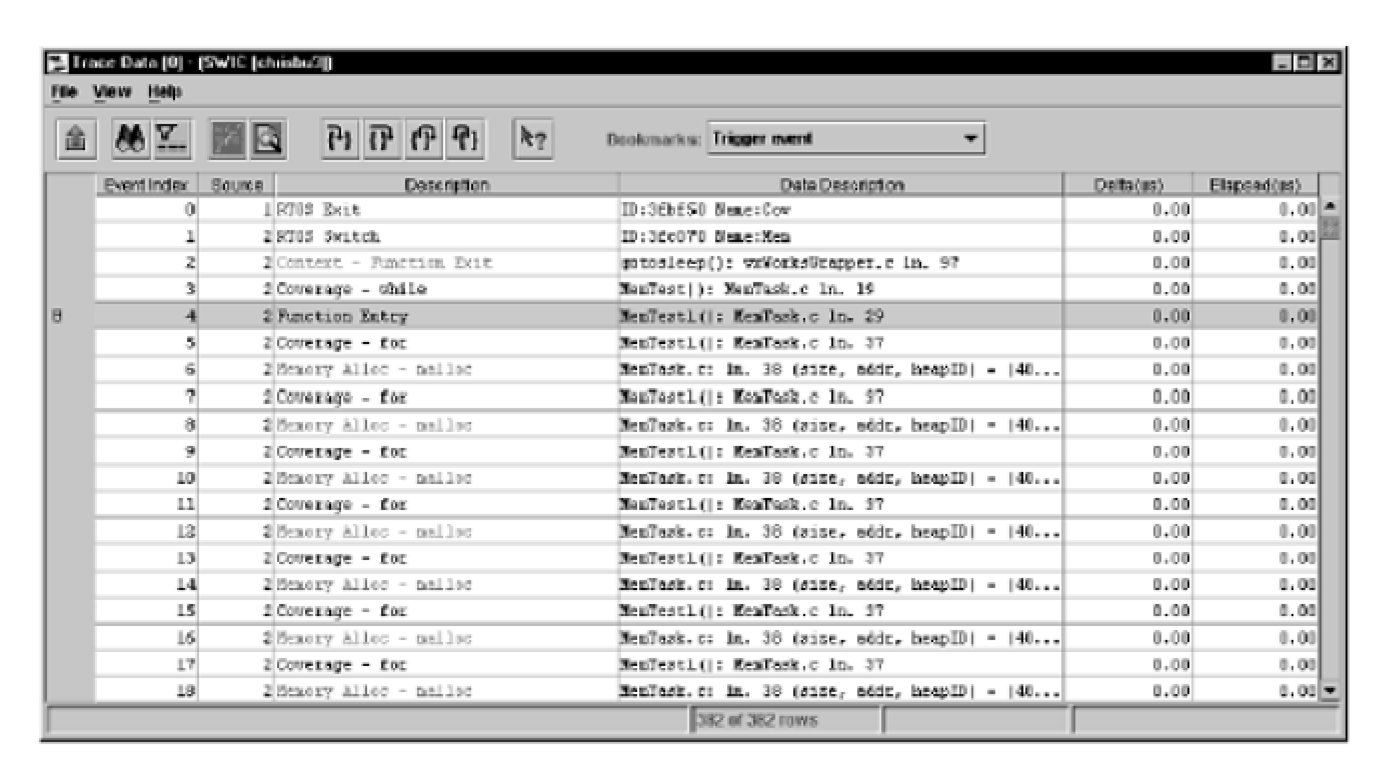


图 3-8　执行追踪分析图

注：图 3-5～图 3-8 摘自 Webmaster 中《白盒测试工具 Code Test》的图。

3.5.2　静态测试和静态测试工具

1. 静态测试

静态测试是测试中重要的手段之一，它可以对各种软件文档进行测试，是软件开发中十分有效的质量控制方法之一。静态测试方法可分为人工测试方法和计算机辅助静态分析方法。静态测试大约可找出 30%～70%的逻辑设计错误。静态测试的重点如下：

- 编码的规范性。
- 资源是否释放。
- 数据结构是否完整和正确。
- 是否有死代码和死循环。
- 代码本身是否存在明显的效率和性能问题。
- 代码本身的方法、类和函数的划分是否清晰且易理解。
- 代码本身是否健壮，是否有完善的异常处理和错误处理。

这里主要介绍人工测试方法。人工测试不要求在计算机上实际执行所测程序，可发挥人的逻辑思维优势，通过桌面检查对软件进行分析，桌面检查包括走读、走查、代码的静态分析、审查和评审。

（1）走读

可由开发人员和测试人员相互走读代码。走读代码主要具有检查文档和源程序代码、检查项目、检查功能、检查界面、检查流程、检查提示信息、检查函数、检查数据类型与变量、检查条件判断、检查循环、检查输入/输出、检查注释、检查程序（模块）、检查数据库等 14 点内容。

检查文档和源程序代码时需要注意以下问题：

- 一份最新的设计文档。
- 程序结构图。
- 所有的模块源程序代码。
- 代码体系结构描述。
- 目录文件。
- 代码组织。

检查项目时需要注意以下问题：

- 检查变量的交叉引用表，重点是检查未说明的变量和违反了类型规定的变量。
- 对照源程序，逐个检查变量的引用、变量的使用序列。
- 临时变量在某条路径上的重写情况。
- 局部变量、全局变量与特权变量的使用。
- 检查标号的交叉引用表。
- 验证所有标号的正确性。
- 检查所有标号的命名是否正确。
- 转向指定位置的标号是否正确。
- 检查子程序、宏、函数，验证每次调用与所调用位置是否正确。
- 确认每次所调用的子程序、宏、函数是否存在。
- 检验调用序列中调用方式与参数顺序、个数、类型上的一致性。
- 检查全部等价变量的类型的一致性，解释所包含的类型差异。
- 确认每个常量的取值、数制、数据类型。
- 检查常量每次引用同它的取值、数制和类型的一致性。
- 用标准检查程序或手工检查程序中违反标准的问题。
- 检查在程序设计风格方面发现的问题。

检查功能时需要注意以下问题：

- 重复的功能。
- 多余的功能。
- 功能实现与设计要求是否相符。
- 功能的使用性、方便性、易用性。

检查界面时需要注意以下问题：

- 界面是否美观。
- 控件的排列、格式。
- 焦点控制是否合理或全面。

检查流程时需要注意以下问题：

- 流程控制是否符合要求。
- 流程实现是否完整。

检查提示信息时需要注意以下问题：

- 提示信息出现时机的合理性。
- 提示信息的格式和要求的合理性。
- 提示框返回后停留位置的合理性。

检查函数时需要注意以下问题：

- 函数头能否清楚地描述函数和它的功能。
- 代码中是否有相关注解。
- 函数名是否能清晰地定义它的目标以及函数的功能。
- 函数的参数是否都被使用。
- 函数的参数接口关系是否清晰。
- 函数的出口是否都有返回值。
- 函数的异常处理是否清楚。

检查数据类型与变量时需要注意以下问题：

- 数据有效性检测是否合理。
- 数据来源的正确性。
- 数据处理过程的正确性。
- 数据处理结果的正确性。
- 是否提供数据类型的解释。
- 变量是否分配了正确的长度、类型和存储空间。
- 是否明确区分静态变量。
- 变量是否初始化。
- 变量的命名是否与标准库中的命名相冲突。
- 是否有对全局变量的描述。
- 类型转换是否正确。

检查条件判断时需要注意以下问题：

- If/Else 使用是否正确。
- 无嵌套的 If 是否使用正确。
- 数字、字符、指针和 0/Null/False 判断是否明确。
- 是否存在臃肿的判断逻辑。
- 所有的判断条件边界是否正确。
- 判断体是否足够短。

检查循环时需要注意以下问题：

- 循环体是否为空。
- 循环之前是否存在初始化代码。
- 有明确的多次循环操作，尽量使用 For 循环。
- 不明确的多次循环操作，尽量使用 While 循环。

- 循环终止的条件是否清晰。
- 所有的循环边界是否正确。
- 循环体内的循环变量是否起到指示作用。

检查输入/输出时需要注意以下问题：

- 所有文件的属性描述是否清楚。
- 输入参数的异常是否存在处理程序。
- 检查文件结束的条件。

检查注释时需要注意以下问题：

- 是否存在一个简单的说明，用于描述代码的结构。
- 每个文件和模块是否均给予解释。
- 解释说明是否能准确描述代码意义。
- 解释是否过于简单。
- 注解是否正确。
- 代码的注释与代码是否一致，注释是否是多余的。

检查程序（模块）时需要注意以下问题：

- 程序中所有的异常是否处理了。
- 程序中是否存在重复的代码。
- 程序结构是否清晰。

检查数据库时需要注意以下问题：

- 数据库命名需要使用小写英语字母、数字和下划线，无其他字符。
- 数据库命名采用项目名或产品名称命名，长度小于 20 位。
- 数据库中的所有表字符集需要统一。
- 数据库对象的命名不使用保留关键字。
- 数据库设计考虑到将来可能存在的异种数据库迁移。
- 字段与界面项目能够一一对应（部分标识符字段和系统设定字段除外）。
- 字段取值需要符合域定义。
- 字段的类型和长度能够满足字段的值的最大限量。
- 文本字段有充足的余量对应可能的长度变更。
- 数字字段考虑了充足的余量和精度对应可能的长度或精度变更。

（2）走查

走查是一种非正式的评审过程。在此过程中，设计者或程序员事先读过设计和编码。由其他成员提出问题并对有争议的部分评论。检查的要点是代码编写是否符合标准和规范、是否存在逻辑错误。

（3）代码的静态分析

代码的静态分析对开发人员的技能要求很高，在没有实际运行程序时就能够清楚地知道程序

潜在的漏洞和缺陷。

代码的静态分析可分为静态结构分析和静态质量度量等。

① 静态结构分析

静态结构分析主要是检查以下几项。

- 审核代码风格和规则。
- 审核程序设计和结构。
- 审核业务逻辑。
- 对表达式进行分析以发现和纠正在表达式中出现的错误，如在表达式中不正确地使用了括号造成错误、数组下标越界错误、除数为零错误、浮点数计算的误差等。
- 对接口的一致性进行分析，如各模块之间接口的一致性；模块与外部数据库的接口一致性；形参与实参在类型数量、顺序、维数、使用上的一致性；全局变量和公共数据区在使用上的一致性。
- 检查函数的调用关系是否正确、是否存在孤立的函数而没有被调用、明确函数被调用的频繁度、对调用频繁的函数可以重点检查。
- 审核模块控制流图，模块控制流图是由许多结点和连接结点的边组成的图形，其中每个结点代表一条或多条语句，边表示控制流向，可以直观地反映出一个函数的内部结构。

② 静态质量度量

静态质量度量可分为软件质量和质量度量模型两方面考虑。软件质量根据 ISO/IEC 9126 国际标准，主要讨论功能性、可靠性、可用性、有效性、可维护性、轻便性。

（4）审查

审查是一种正式的检查和评估方法，它是用逐步检查源代码中有无逻辑或语法错误的办法来检查故障。审查内容主要包括以下几项：

- 检查代码和设计的一致性。
- 检查代码对标准的遵循、可读性。
- 检查代码的逻辑表达的正确性。
- 检查代码结构的合理性。

（5）评审

评审通常是在审查后进行的，根据编码检查单和设计相关工件对编码进行评审。这里可以由开发人员自己对代码进行讲解，也可以由他人对编码进行讲解。

评审分为两个步骤：

- 小组负责人提前把设计规格说明书、控制流程图、程序文本及有关要求、规范等分发给小组成员阅读。
- 召开程序审查会，在会上，首先由程序员逐句讲解程序的逻辑，在此过程中，程序员或其他小组成员可以提出问题，展开讨论，审查错误是否存在。

2. 静态测试工具

静态测试工具是用计算机辅助静态分析方法，直接对代码进行分析，不需要运行代码，也不需要对代码编译链接，生成可执行文件。静态测试工具一般是对代码进行语法扫描，找出不符合编码规范的地方，根据某种质量模型评价代码的质量，生成系统的调用关系图等。

静态测试工具的代表有：支持编辑环境和编译器的 PC-Lint；Mercury Interactive 公司的 WinRunner 企业级功能测试工具；Telelogic 公司的 Logiscope 软件；PR 公司的 PRQA 软件；CheckMate、QAC++、QStudio for Java 等。

3.5.3 动态测试和动态测试工具

1. 动态测试

动态测试主要应用在针对程序的内部功能进行，检测程序功能是否符合设计要求，通过选择适当的测试用例，实际运行所测程序，比较实际运行结果和预期结果，动态测试把被测代码放在相对真实的环境下运行，并且从多角度观察程序运行时能体现的功能、逻辑、行为、结构等行为，以发现其中的错误。

动态测试的基本特征：通过运行软件来检验软件的动态行为和运行结果的正确性。

动态测试的流程如下：

- 选取定义域有效值，或定义域外无效值。
- 对已选取值决定预期的结果。
- 用选取值执行程序。
- 执行结果与对已选取值决定预期的结果相比不吻合，则程序有错。

（1）动态测试原则

动态测试具有以下原则：

- 保证每个模块的所有独立路径至少被使用一次。
- 对所有的逻辑值均测试 True 和 False。
- 上下边界及可操作范围内运行所有循环。
- 检查内部数据结构以确保其有效性。

（2）动态测试方法

动态测试方法分为结构测试和正确性测试：

- 结构测试采用语句测试、分支测试或路径测试。
- 正确性测试（功能性测试）是基于产品功能规格说明书、从用户角度针对产品特定的功能和特性所进行的验证活动，以确认每个功能是否得到完整的实现、用户能否正常使用这些功能。功能测试一般要在完成集成测试后进行，而且是针对应用系统、在实际运行环境下而进行的测试。

（3）动态测试的三个程序

动态测试的三个程序分别为测试覆盖监视程序、断言处理程序、符号执行程序。

① 测试覆盖监视程序

测试覆盖监视程序主要用在结构测试中，用于监视测试的实际覆盖程度。主要的工作如下：

- 分析并输出每一可执行语句的执行特性。
- 分析并输出各分支或各条路径的执行特性。
- 计算并输出程序中谓词的执行特性。

测试覆盖监视程序的工作过程可分为以下三个阶段：

- 对所测试程序进行预处理，如在程序的分支点和汇合点插入“执行计数探针”；在非简单赋值语句（相对于赋常数值或下标计算等简单赋值语句而言）后插入“记忆变量值探针”，记录变量的首次赋值、末次赋值、最小值、最大值，以及在循环语句中插入的“记忆变量值探针”，用于记录循环控制变量的首次赋值、末次赋值、最小值、最大值。
- 编译预处理后的源程序，运行目标程序。在运行过程中，利用探针监视、检查程序的动态行为，收集与统计的相关信息。
- 一组测试后，可以根据要求，输出某一语句的执行次数、某一转移发生的次数、某赋值语句的数值范围、某循环控制变量的数据范围、某子程序运行的时间和所调用次数等，从而发现在程序中从未执行的语句、不应该执行而实际执行了的语句、应该执行但实际没有执行的语句，以及发现不按预定要求终止的循环、下标值越界、除数为零等异常情况。

② 断言处理程序

“断言”是指变量应满足的条件，用注释语句写出的断言叫做断言语句，断言条件如 k<8、B(6)>4 等。在所测试的源程序中，程序执行时对照断言语句检查事先指定的断言是否成立，从而帮助复杂系统的检验、调试和维护。

断言可分为局部性断言和全局性断言两类。

- 局部性断言是指在程序的某一位置上，如重要的循环/过程的入口和出口处，或者在一些可能引起异常的关键算法之前设置的断言语句。
- 全局性断言是指在程序运行过程中自始至终都适用的断言。

动态断言处理程序的工作过程如下：

- 动态断言处理程序对语言源程序进行预处理，为注释语句中的每一个断言插入一段相应的检验程序。
- 运行经过预处理的程序，检验程序实际运行结果与断言所规定的逻辑状态是否一致。对于局部性断言，每当程序执行到这个位置时，相应的检验程序就要工作；对于全局性断言，在每次变量被赋值后，相应的检验程序就进行工作。
- 动态断言处理程序还要统计检验的结果（即断言成立或不成立的次数），在发现断言不成立时，还要记录当时的现场信息，如有关变量的状态等。处理程序还可按测试人员的要求，在某个断言不成立的次数已达指定值时中止程序的运行，并输出统计报告。

- 一组测试结束后，程序输出统计结果、现场信息，以供测试人员分析。

③ 符号执行程序

符号执行程序是一种介于程序测试用例执行与程序正确性证明之间的方法。它使用了一个专用的程序，对输入的源程序进行解释。在解释执行时，所有的输入都以符号形式输入到程序中，这些输入包括基本符号、数字及表达式等。符号执行的结果可以有以下两个用途：

- 检查公式的执行结果是否达到程序预期的目的。
- 通过程序的符号执行产生程序的路径，为进一步自动生成测试数据提供条件。

符号执行程序的最重要用途是产生测试数据。

2. 动态测试工具

动态测试工具与静态测试工具不同，动态测试工具一般采用“插桩”的方式，向代码生成的可执行文件中插入一些监测代码，用来统计程序运行时的数据。其与静态测试工具最大的不同在于动态测试工具要求被测系统实际运行。动态测试工具的代表有：Compuware 公司的 DevPartner 软件；Rational 公司的 Purify 系列、Robot、GUI、QACenter、WinRunner 等。

第4章　黑盒测试技术

黑盒测试技术是软件测试的主要方法之一，黑盒测试的方法、工具、操作步骤等内容是必须掌握的，本章重点讨论以下内容：

- 黑盒测试的基本概念。
- 黑盒测试的方法。
- 黑盒测试的工具。
- 黑盒测试的操作步骤。

4.1　黑盒测试的基本概念

黑盒测试又称为数据驱动测试或基于规范的测试。利用这种方法进行测试时，可把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，注重于测试软件的功能性需求，测试者在程序接口处进行测试，只检查程序功能是否按照规格说明书的规定正常使用，程序是否能接收输入数据而产生正确的输出信息，并且保持数据库或文件的完整性。依据程序功能的需求规范考虑确定测试用例和推断测试结果的正确性。它是在已知产品所应具有的功能，通过测试来检测每个功能是否都能正常运行，因此黑盒测试是从用户观点出发的测试。

由于黑盒测试不需要了解程序内部结构，所以许多高层的测试，如确认测试、系统测试、验收测试都采用黑盒测试。

黑盒测试有两种结果，即通过测试和测试失败。如果规格说明有误，则利用黑盒测试方法是无法发现的。

黑盒测试能发现以下几类错误：

- 功能不对或遗漏。
- 界面错误。
- 数据结构或外部数据库访问错误。
- 性能错误。
- 初始化和终止错误。

黑盒测试对程序的功能性测试具有以下要求：

- 每个软件特性必须被一个测试用例或一个被认可的异常所覆盖。
- 利用数据类型和数据值的最小集测试。
- 利用一系列真实的数据类型和数据值运行，测试超负荷及其他“最坏情况”的结果。
- 利用假想的数据类型和数据值运行，测试排斥不规则输入的能力。

- 测试影响性能的关键模块，如基本算法、精度、时间、容量等是否正常。

4.1.1 黑盒测试的优点和缺点

1. 黑盒测试的优点

黑盒测试具有如下优点：

- 有针对性地找问题，并且定位问题更准确。
- 黑盒测试可以证明产品是否达到用户要求的功能，符合用户的工作要求。
- 能重复执行相同的动作，测试工作中最枯燥的部分可交由机器完成。

2. 黑盒测试的缺点

黑盒测试具有如下缺点：

- 需要充分了解产品用到的技术，测试人员需要具有较多经验。
- 在测试过程中很多是手工测试操作。
- 测试人员要负责大量文档、报表的制订和整理工作。

4.1.2 黑盒测试与白盒测试的比较

黑盒测试与白盒测试的区别主要存在以下 3 个方面中。

1. 已知产品的因素

在已知产品的因素方面具有以下区别。

- 黑盒测试：已知产品的功能设计规格，可以通过测试证明每种实现的功能是否符合要求。
- 白盒测试：已知产品的内部工作结构，可以通过测试证明每种内部操作是否符合设计规格要求，以及所有内部成分是否经过检查。

2. 检查测试的主要内容

黑盒测试检查的主要内容如下：

- 是否有不正确或遗漏的功能？
- 在接口上，输入是否能正确的接受？能否输出正确的结果？
- 是否有数据结构错误或外部信息（例如数据文件）访问错误？
- 功能上是否能够满足要求？
- 是否有初始化或终止性错误？

白盒测试检查的内容如下：

- 对程序模块的所有独立的执行路径至少测试一遍。
- 对所有的逻辑判定，取“真”与取“假”的两种情况都能至少测试一遍。
- 在循环的边界和运行的界限内执行循环体。
- 测试内部数据结构的有效性等。

3. 静态测试方法

在静态测试方法方面，白盒测试与黑盒测试具有以下区别。

- 静态白盒测试方法：走查、复审、评审程序源代码、数据字典、系统设计文档、环境设置、软件配置项等。
- 静态黑盒测试方法：文档测试，特别是产品需求文档、用户手册、帮助文件等的审查。

4. 动态测试方法

在动态测试方法方面，白盒测试与黑盒测试具有以下区别。

- 动态白盒测试方法：通过驱动程序来调用，如进行单元测试、集成测试和部分性能（或可靠性、恢复性）测试等。
- 动态黑盒测试方法：通过数据输入并运行程序来检验输出结果，如功能测试、验收测试和一些性能（或兼容性、安全性）测试等。

4.2　黑盒测试的方法

由于开发的速度比较快，用户的需求多变，需要不断地调整应用，所以要求对软件有更加严格的测试。由于不断变化的需求将导致应用不同版本的产生，每一个版本都需要对它测试，测试工作头绪多，测试人员难以组织科学、全面的测试用例，从而影响测试的全面性和有效性。测试过程要求大量因素的配合，包括许多的步骤、测试者、大量测试数据和不同应用的多种版本等。而黑盒测试着眼于程序的外部结构，不考虑内部逻辑结构，针对软件界面和软件功能进行测试，黑盒测试人员需要一个快速、可重用的测试过程，从而最有效地使用现有测试资料、测试方法和应用测试工具建立测试用例，自动执行测试和产生文档结果。

采用黑盒技术设计测试用例的方法主要有以下几种：

- 等价类划分方法。
- 边界值分析方法。
- 错误推测方法。
- 因果图方法。
- 判定表驱动分析方法。
- 正交实验设计方法。
- 功能图分析方法。
- 场景设计方法。

4.2.1　等价类划分方法

1. 划分等价类的方法简述

等价类划分法是把程序的输入域划分成若干部分，然后从每个部分中选取少数的代表性数据当作测试用例。等价类划分方法是一种重要的、常用的黑盒测试用例设计方法，利用这一方法设计测试用例可以不用考虑程序的内部结构，即以需求规格说明书为依据，仔细分析和推敲说明书的各

项需求，特别是功能需求，把说明中对输入的要求和输出的要求区别开来并加以分解。

等价类是指某个输入域的子集合。每一子集合代表每一类，每一类的代表性数据在测试中的作用都是等效于这一类中的其他值，也就是说，测试某等价类的代表值就等于对这一类其他值的测试。如果这一类中的一个例子发现了错误，这一等价类中的其他例子也能发现同样的错误；反之，如果某一类中的一个例子没有发现错误，则这一类中的其他例子也不会查出错误。这就可以用少量代表性的测试数据，取得较好的测试结果。使用这一方法设计测试用例，必须在分析需求规格说明、功能说明的基础上找出每个输入条件，然后为每个输入条件划分两个或多个等价类，列出等价类表。等价类可分为有效等价类和无效等价类。

（1）有效等价类

有效等价类是指由对于程序的规格说明来说是合理的、有意义的输入数据所构成的集合。在具体项目中，有效等价类可以是一个，也可以是多个。利用有效等价类可检验程序是否实现了规格说明中所规定的功能和性能。

（2）无效等价类

与有效等价类相反，无效等价类是指由对程序的规格说明不合理或无意义的输入数据所构成的集合。在具体项目中，无效等价类至少应有一个，也可能有多个。

利用等价类设计测试用例时，要同时考虑这两种等价类，因为软件不仅要能接收合理的数据，也要接收不合理的数据进行检验。这样的测试才能确保软件具有更高的可靠性。

2. 划分等价类的原则

划分等价类主要有如下原则：

- 在输入条件规定了取值范围或值的个数的情况下，则可以确立一个有效等价类和两个无效等价类。
- 在输入条件规定了输入值的集合或者规定了“必须如何”的条件的情况下，可确立一个有效等价类和一个无效等价类。
- 在输入条件是一个布尔量的情况下，可确定一个有效等价类和一个无效等价类。
- 在规定了输入数据的一组值（假定 n 个），并且程序要对每一个输入值分别处理的情况下，可确立 n 个有效等价类和一个无效等价类。
- 在规定了输入数据必须遵守的规则的情况下，可确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）。
- 在确定已划分的等价类中各元素在程序处理中的方式不同的情况下，则应再将该等价类进一步地划分为更小的等价类。

3. 划分等价类的要求

划分等价类的要求如下：

- 测试完备合理、避免冗余。
- 划分输入条件、有效等价类和无效等价类时最重要的是：将集合划分为互不相交的一组子集。
- 整个集合完备。

- 子集互不相交，保证一种形式的无冗余性。
- 同一类中标识（选择）一个测试用例，在同一等价类中，往往处理相同，相同处理映射到“相同的执行路径”。

4. 等价类表的建立

在建立等价类后，可建立等价类表，从而列出所有划分出的等价类。等价类表的建立如表 4-1 所示。

表 4-1　等价类表的建立

输入条件	有效等价类	无效等价类

等价类表建立后，从划分出的等价类中按以下步骤确定测试用例。

- 为每一个等价类规定一个惟一的编号。
- 设计一个新的测试用例，使其尽可能多地覆盖尚未被覆盖地有效等价类，重复这一步，直到所有的有效等价类都被覆盖为止。
- 设计一个新的测试用例，使其仅覆盖一个尚未被覆盖的无效等价类，重复这一步，直到所有的无效等价类都被覆盖为止。

5. 测试用例等价类表的建立

【例 1】　建立小区物业住宅管理系统“日期检查功能”的测试用例等价类表。

有一个小区物业住宅管理系统，要求住户输入以年月表示的日期。假设日期限定在 1998 年 1 月~2068 年 12 月，并规定日期由 6 位数字字符组成，前 4 位表示年，后 2 位表示月。现用等价类划分法设计测试用例，“日期检查功能”的测试用例等价类表如表 4-2 所示。

表 4-2　“日期检查功能”的测试用例等价类

输入等价类	有效等价类	无效等价类
日期的类型及长度	①6 位数字字符	④有非数字字符 ⑤少于 6 位数字字符 ⑥多于 6 位数字字符
年份范围	②在 1998~2068 之间	⑦小于 1998 ⑧大于 2068
月份范围	③在 01~12 之间	⑨等于 00 ⑩大于 12

6. 测试用例的设计

【例 2】　【例 1】的小区物业住宅管理系统“日期检查功能”的测试用例设计。

覆盖所有的有效等价类，在表 4-2 中列出了 3 个编号，分别为①、②、③；覆盖所有的无效等价类，在表 4-2 中列出了 7 个编号，分别为④、⑤、⑥、⑦、⑧、⑨、⑩，设计一个测试用例，设计的测试用例结果如表 4-3 所示。

表 4-3　设计的测试用例结果

测试数据	期望结果	覆盖的有效/无效等价类
200611	输入有效	①、②、③
199901	输入有效	①、②、③
205901	输入有效	①、②、③
9954￥9	无效输入	④
20096	无效输入	⑤
20120607	无效输入	⑥
198901	无效输入	⑦
200401	无效输入	⑧
200400	无效输入	⑨
200422	无效输入	⑩

4.2.2　边界值分析方法

1. 边界值分析方法简述

边界值分析法 BVA（Boundary Value Analysis）用于列出单元功能、输入、状态及控制的合法边界值和非法边界值，对数据进行测试，检查用户输入的信息、返回结果以及中间计算结果是否正确。补充等价划分的测试用例设计技术。边界值分析法比较简单，仅是用于考察正处于等价划分边界或在边界附近的状态，选择输入和输出等价类的边界，选取正好等于、刚刚大于或刚刚小于边界的值作为测试数据，而不是选取等价类中的典型值或任意值作为测试数据。它是对等价类划分方法的补充，不仅重视输入条件边界，而且也从输出域中导出测试用例。典型的边界值分析包括 IF 语句中的判别值、定义域、值域边界、空或畸形输入等。边界值分析法是以边界情况的处理作为主要目标专门设计测试用例的方法。

2. 选择边界值的设计原则

对边界值设计测试用例，应遵循以下原则：

- 如果输入条件规定了值的范围（或是规定了值的个数），则应取刚达到这个范围的边界的值，以及刚刚超越这个范围边界的值作为测试输入数据。
- 如果输入条件规定了值的个数，则用最大个数、最小个数、比最小个数少一、比最大个数多一的数作为测试数据。
- 如果程序的规格说明给出的输入域或输出域是有序集合，则应选取集合的第一个元素和最后一个元素作为测试用例。
- 如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构的边界上的值作为测试用例。
- 分析规格说明，找出其他可能的边界条件。

3. 常见的边界值

常见的边界值如下：

- 屏幕上光标在最左上、最右下位置。
- 报表的第一行和最后一行。
- 数组元素的第一个和最后一个。
- 循环的第 0 次、第 1 次、倒数第 2 次、最后一次。

测试所包含的边界检验有几种类型，如数字、字符、位置、大小、方位、尺寸、空间等。

4.2.3　错误推测方法

错误推测方法的基本思想是：利用直觉和经验猜测出出错的可能类型，列举出程序中所有可能的错误和容易发生错误的情况，基本思想是列举出可能犯的错误或错误易发情况的清单，然后依据清单来编写测试用例；并且在阅读规格说明时联系程序员可能做的假设来确定测试用例。这种方法在很大程度上是凭经验进行的，即凭借人们对过去所作测试工作结果的分析，对所揭示的缺陷的规律性作直觉的推测来发现缺陷。

4.2.4　判定表驱动分析方法

1. 判定表驱动分析方法简述

判定表（Decision Table）是分析和表达多逻辑条件下执行不同操作情况的工具，能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏，因此，利用判定表设计的测试用例集合称为判定表驱动分析方法。在一些数据处理问题中，某些操作的实施依赖于多个逻辑条件的组合，即针对不同逻辑条件的组合值，分别执行不同的操作。判定表很适合于处理这类问题。

2. 判定表的组成

判定表通常由 4 个部分组成，如图 4-1 所示。

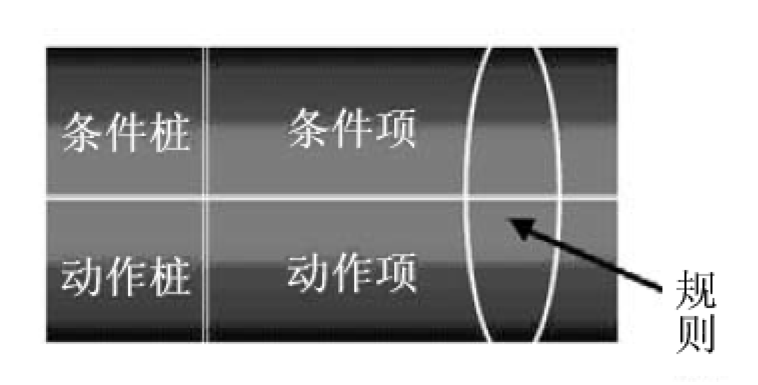


图 4-1　判定表组成的 5 个部分

对图 4-1 的说明如下。

- 条件桩（Condition Stub）：列出了问题的所有条件。通常认为列出的条件的次序无关紧要。
- 动作桩（Action Stub）：列出了问题规定可能采取的操作，这些操作的排列顺序没有约束。
- 条件项（Condition Entry）：列出针对它左列条件的取值。
- 动作项（Action Entry）：列出在条件项的各种取值情况下应该采取的动作。
- 规则：任何一个条件组合的特定取值及其相应要执行的操作称为规则。在判定表中贯穿条

件项和动作项的一列就是一条规则。有 n 个条件，每个条件有两个取值（0，1），故有 2^n 条规则。显然，判定表中列出多少组条件取值，也就有多少条规则。

3. 规则合并

规则合并是指有两条或多条规则合并为一条规则。

合并条件如图 4-2 所示。

- 左端：1、1，合并为右端：1。
- 左端：-、0，合并为右端：-。
- 左端：0、0，合并为右端：0。
- 左端：1、0，合并为右端：1。
- 左端：0、1，合并为右端：1。

无关条件项“-”可包含其他条件项取值，具有相同动作的规则可合并。

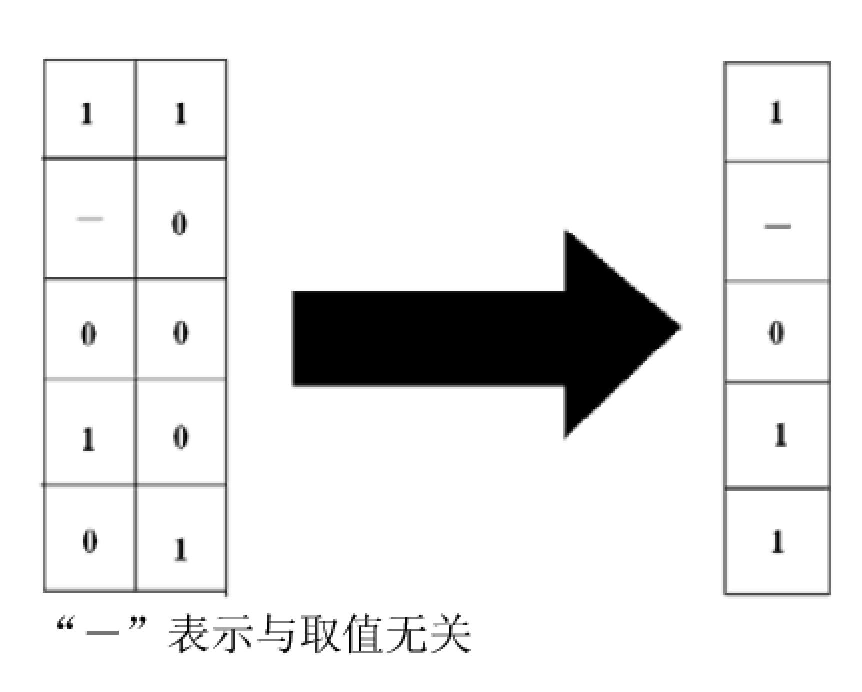


图 4-2　规则合并

4.2.5　因果图方法

1. 因果图方法简述

因果图法是一种适合于描述对于多种条件的组合、相应产生多个动作的形式的方法，利用图解法分析输入的各种组合情况，从而设计测试用例的方法，它适合于检查程序输入条件的各种组合情况。等价类划分方法和边界值分析方法都着重考虑输入条件，但没有考虑输入条件的各种组合、输入条件之间的相互制约关系，虽然各种输入条件可能出错的情况已经测试到了，但多个输入条件组合起来可能出错的情况却被忽视了，要检查输入条件的组合不是一件容易的事情，即使把所有输入条件划分成等价类，它们之间的组合情况也相当多，因此必须考虑采用一种适合于描述多种条件的组合，相应产生多个动作的形式来设计测试用例，这就需要采用因果图。

采用因果图法能帮助我们按照一定的步骤选择一组高效的测试用例，同时，还能指出程序规范的描述中存在什么问题。

因果图法最终生成的是判定表，适合于检查程序输入条件的各种组合情况。

2. 因果图的关系符号和约束

（1）关系符号

① 恒等

恒等关系符号如图 4-3 所示。

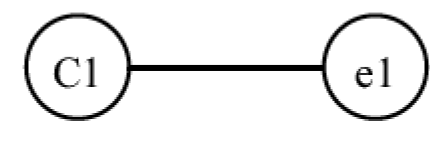


图 4-3　恒等关系符号

② 非

非关系符号如图 4-4 所示。

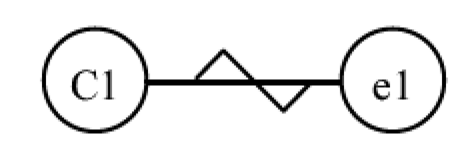


图 4-4　非关系符号图

③ 或

或关系符号如图 4-5 所示。

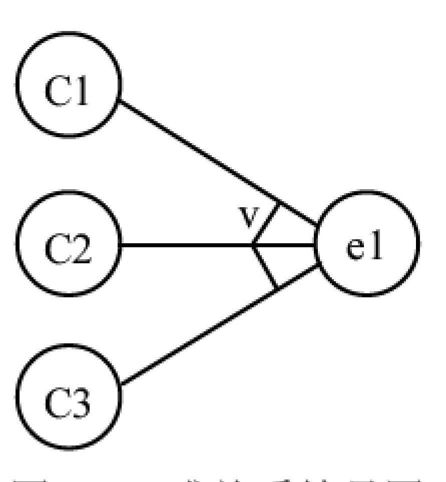


图 4-5　或关系符号图

④ 与

与关系符号如图 4-6 所示。

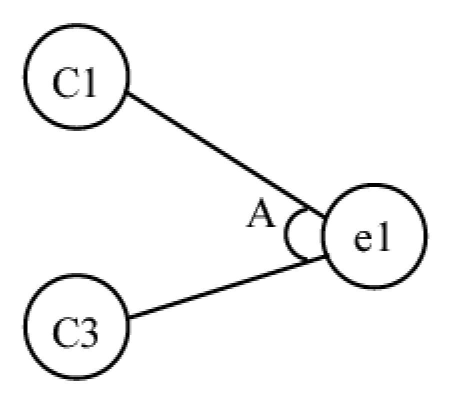


图 4-6　与关系符号图

对图 4-3~图 4-6 的说明如下：

- 因果图中使用了简单的逻辑符号，以直线连接左右结点。左结点表示输入状态（或称原因），右结点表示输出状态（或称结果）。
- Ci 表示原因，通常置于图的左部；ei 表示结果，通常置于图的右部。Ci 和 ei 均可取值 0 或 1，0 表示某状态不出现，1 表示某状态出现。

（2）约束

输入状态之间还可能存在某些依赖关系，这种关系称为约束，例如某些输入条件本身不可能同时出现。输出状态之间也往往存在约束。在因果图中使用特定的符号标明这些约束。

① E约束符号

E 约束符号（异）是指：a 和 b 中至多有一个可能为 1，即 a 和 b 不能同时为 1。E 约束符号如图 4-7 所示。

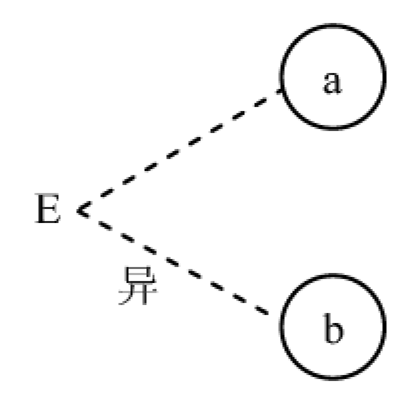


图 4-7　E 约束符号图

② I 约束符号

I 约束符号（或）是指：a、b 和 c 中至少有一个必须是 1，即 a、b 和 c 不能同时为 0。I 约束符号如图 4-8 所示。

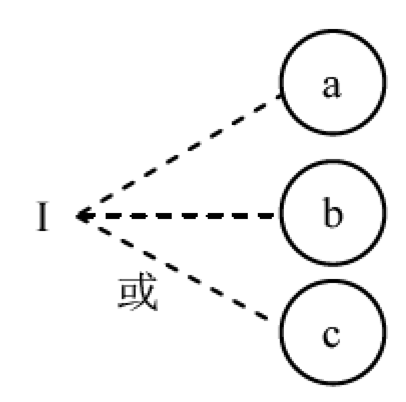


图 4-8　I 约束符号图

③ O 约束符号

O 约束符号（惟一）是指：a 和 b 必须有一个，且仅有一个为 1。O 约束符号如图 4-9 所示。

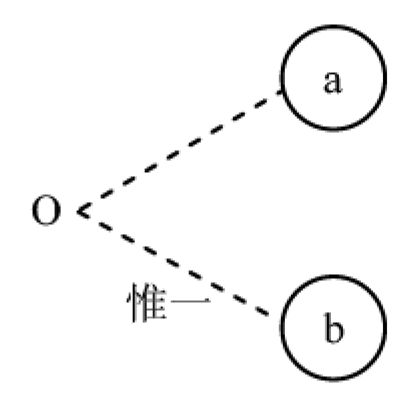


图 4-9　O 约束符号图

④ R 约束符号

R 约束符号（要求）是指：a 是 1 时，b 必须是 1，即不可能 a 是 1 时 b 是 0。R 约束符号如图 4-10 所示。

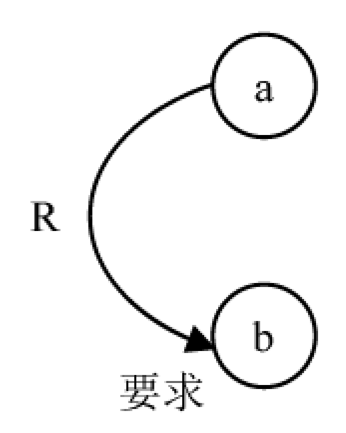


图 4-10　R 约束符号图

⑤ M 约束符号

M 输出条件的约束（强制）是指：若结果 a 是 1，则结果 b 强制为 0。M 约束符号如图 4-11 所示。

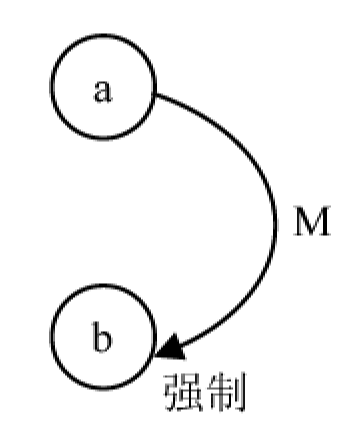


图 4-11　M 约束符号图

3. 利用因果图导出测试用例的基本步骤

利用因果图导出测试用例需要经过以下几个步骤：

- 分析程序规范、规格说明描述中哪些是原因，哪些是结果，原因常常是输入条件或是输入条件的等价类；结果是输出条件，并给每个原因和结果赋予一个标识符。
- 分析程序规范、规格说明描述中语义的内容，找出原因和结果之间、原因和原因之间的关系，根据这些关系画出因果图。
- 在因果图上用一些记号表明约束或限制条件。
- 把因果图转换为判定表。
- 把判定表的每一列拿出来作为依据，设计测试用例。

4. 判定表的组成

因果图方法中的判定表（Decision Table）是分析和表达多逻辑条件下执行不同操作的情况下的工具。在程序设计发展的初期，由于它可以把复杂的逻辑关系和多种条件组合的情况表达的既具体又明确，因此判定表已被当作编写程序的辅助工具。

（1）判定表组成的规则要求

任何一个条件组合的特定取值及其相应要执行的操作，在判定表中贯穿条件项和动作项的一列为规则。显然，判定表中列出多少组条件取值，也就有多少条规则，即条件项和动作项就有多少列。

（2）判定表的建立步骤

判定表的建立步骤如下：

- 确定规则的个数，假如有 n 个条件（原因），每个条件有两个取值（0,1），故有 2^n 种规则。
- 列出所有的条件桩和动作桩。
- 填入条件项。
- 填入动作项。
- 简化、合并相似规则（相同动作）。

【例 3】 某软件规格说明书包含这样的要求：第一列字符必须是 A 或 B，第二列字符必须是一个数字，在此情况下进行文件的修改，如果第一列字符不是 A 或 B，则给出信息 L；如果第二列字符不是数字，则给出信息 M。

根据题意列出如下原因。

- 原因 1：第一列字符是 A。
- 原因 2：第一列字符是 B。
- 原因 3：第二列字符是一数字。

结果如下。

- 结果 21：修改文件。
- 结果 22：给出信息 L3002
- 结果 23：给出信息 M。



11 为中间原因；考虑到原因 1 和原因 2 不可能同时为 1，因此在因果图上施加 E 约束。

对应的因果关系表如表 4-4 所示。

表 4-4 因果关系

编号	原因	编号	结果
1	第一列字符是 A	21	修改文件
2	第一列字符是 B	22	给出信息 L
3	第二列字符是一数字	23	给出信息 M
11	中间原因		

对应的因果关系图如图 4-12 所示。

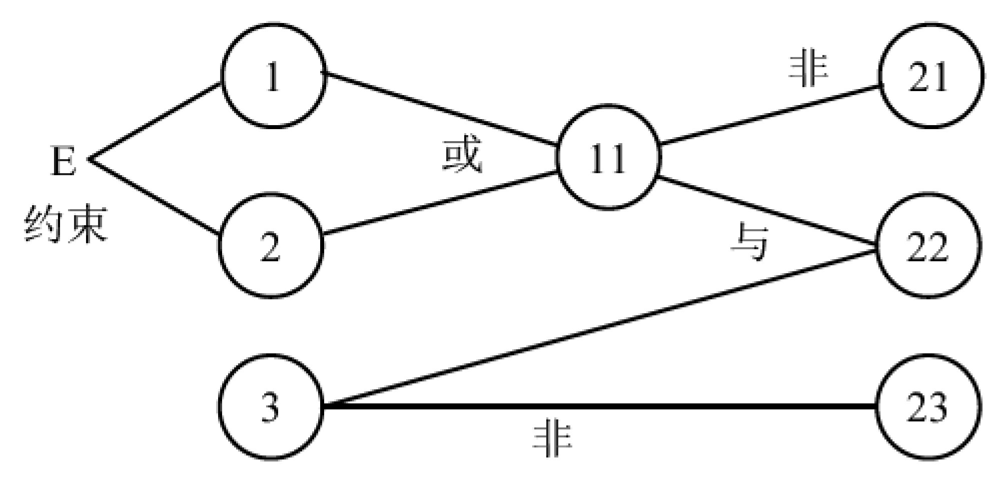


图 4-12 对应的因果关系图

根据因果图建立的判定表如表 4-5 所示。

表 4-5 根据因果图建立的判定

组合情况 / 因/果		组合情况产生对应的动作							
		1	2	3	4	5	6	7	8
原因	1	1	1	1	1	0	0	0	0
	2	1	1	0	0	1	1	0	0
	3	1	0	1	0	1	0	1	0
	11			1	1	1	1	0	0
结果	22			0	0	0	0	1	1
	21			1	0	1	0	0	0
	23			0	1	0	1	0	1
测试用例				Y	Y	Y	Y	Y	Y

对表 4-5 的说明如下。

- Y：测试用例。表的最下一栏给出 6 种情况的测试用例。
- 3 列：结果 21=1。
- 4 列：结果 23=1。
- 5 列：结果 21=1。
- 6 列：结果 23=1。
- 7 列：结果 22=1。
- 8 列：结果 22=1，结果 23=1。

按条件的各种组合情况产生对应的动作，在组合产生对应动作的 8 种情况中，原因 1 和原因 2 同时为 1 是不可能出现的，故应排除这两种情况。

4.2.6 正交实验设计方法

正交实验设计方法是依据 Galois 理论，从大量的（实验）数据（测试例）中挑选适量的、有代表性的点（例），从而合理地安排实验（测试）的一种科学实验设计方法。

正交实验设计方法是使用已经创建的正交表格来安排试验并进行数据分析的一种方法，目的是用最少的测试用例达到最高的测试覆盖率。

利用正交实验设计测试用例的步骤如下。

- 提取功能说明，构造因子状态表：把影响实验指标的条件称为因子，而影响实验因子的条件称为因子的状态，利用正交实验设计方法来设计测试用例时，首先要根据被测试软件的规格说明书找出影响其功能实现的操作对象和外部因素，把它们当作因子，而把各个因子的取值当作状态，对软件需求规格说明中的功能要求进行划分，把整体的概要性的功能要求进行层层分解与展开，分解成具体的具有相对独立性的、基本的功能要求，这样就可以把被测试软件中所有的因子都确定下来，并为确定各因子的权值提供参考依据，确定因子与状态是设计测试用例的关键，因此要求尽可能全面、正确的确定取值，以确保测试用例的设计完整、有效。

- 加权筛选，生成因素分析表：对因子与状态的选择可按其重要程度分别加权，可根据各个因子及状态的作用大小、出现频率的大小、测试的需要确定权值的大小。
- 利用正交表构造测试数据集。

4.2.7 功能图分析方法

功能图分析方法是用功能图 FD 形式化地表示程序的功能说明，由状态迁移图和布尔函数组成。状态迁移图用状态和迁移来描述，一个状态指出数据输入的位置（或时间），而迁移则指明状态的改变，同时要依靠判定表或因果图表示的逻辑功能，并机械地生成功能图的测试用例。

功能图模型由状态迁移图和逻辑功能模型构成：

- 状态迁移图用于表示输入数据序列以及相应的输出数据，在状态迁移图中，由输入数据和当前状态决定输出数据和后续状态。
- 逻辑功能模型用于表示在状态中输入条件和输出条件之间的对应关系，逻辑功能模型只适合于描述静态说明，输出数据仅由输入数据决定。

功能图分析方法中需要用到逻辑覆盖和路径测试的概念及方法，这部分内容属于白盒测试方法的范畴，这里不再赘述。

4.2.8 场景设计方法

现在的软件几乎都是利用事件触发来控制流程的，事件触发时的情景便形成了场景，而同一事件不同的触发顺序和处理结果就形成了事件流。这种在软件设计方面的思想也可以引入到软件测试中，可以比较生动地描绘出事件触发时的情景，有利于测试设计者设计测试用例，同时使测试用例更容易理解和执行。

1. 场景法的基本流和备选流

场景用来描述流经用例的路径，从用例开始到结束遍历这条路径上所有的基本流和备选流，如图 4-13 所示，图中经过用例的每条路径都用基本流和备选流来表示，直黑线表示基本流，是经过用例的最简单路径；备选流用不同的色彩表示，一个备选流可能从基本流开始，在某个特定条件下执行，然后重新加入基本流中（如备选流 1 和 3）；也可能起源于另一个备选流（如备选流 2），或者终止用例而不再重新加入到某个流（如备选流 2 和 4）。



图 4-13 场景法的基本流和备选流

2. 场景法的设计步骤

场景法的设计步骤如下：

01 根据说明描述出程序的基本流及各项备选流。
02 根据基本流和各项备选流生成不同的场景。
03 对每一个场景生成相应的测试用例。
04 对生成的所有测试用例重新复审，去掉多余的测试用例，测试用例确定后，对每一个测试用例确定测试数据值。

4.3　黑盒测试的工具

目前，用于黑盒测试的工具软件有很多，针对不同架构软件的工具也不断推陈出新，这里重点介绍 QACenter 和 WinRunner 测试工具。

4.3.1　QACenter 测试工具

QACenter 测试工具自动帮助测试者管理测试过程、快速分析和调试程序，包括针对回归、强度、单元、并发、集成、移植、容量和负载建立测试用例，自动执行测试和产生文档结果。

QACenter 主要包括以下几个模块。

- QARun：功能测试模块。
- QALoad：性能测试模块。
- EcoTools：可用性管理模块。
- EcoScope：性能优化模块。
- TESTBytes：测试数据自动生成模块。

1. 功能测试模块

在 QACenter 测试工具中，功能测试模块 QARun 主要用于客户/服务器应用客户端的功能测试，主要包括对应用的 GUI（图形用户界面）的测试及客户端事物逻辑的测试。QARun 组件的测试实现方式是通过鼠标移动、键盘点击操作实现，即而得到相应的测试脚本，对该脚本可以进行编辑和调试。在记录的过程中可针对被测应用中所包含的功能点建立期望值，也就是说，在插入检查点的同时建立期望值。检查点是目标系统的一个特殊方面在一特定点的期望状态，通常，检查点在 QARun 提示目标系统执行一系列事件之后被执行。检查点用于确定实际结果与期望结果是否相同。

2. 性能测试模块

性能测试模块 QALoad 是企业范围的负载测试工具，性能测试模块支持的范围广，测试的内容多，可以帮助软件测试人员、开发人员和系统管理人员对于分布式的应用执行有效的负载测试。负载测试能够模拟大批量用户的活动，从而发现在大量用户负载下对 C/S 系统的影响。

性能测试模块的主要特点如下。

- 操作简便：测试人员只需操作被测应用、执行性能关键的事物处理，然后在 QALoad 脚本

中通过服务器上应用调用的需求类型开发这些事物处理即可。

- 广泛的适用性：QALoad 支持 DB2、DCOM、ODBC、Oracle、NETLoad、Corba、QARun、SAP、SQL Server、Sybase、Telnet、TUXEDO、UNIFACE、WinSock、WWW 等。

3. 可用性管理模块

可用性管理模块 EcoTools 提供一个广泛范围打包的 Agent 和 Scenarios，可以立即在测试或生产环境中激活、计划和管理以商务为中心的可用性，EcoTools 支持一些主流成型的应用，如 SAP、PeopleSoft、Baan、Oracle、Uniface 和 LotusNotes 以及定制的应用。可用 EcoTools 监控服务器性能和服务器资源，尤其是监控 Windows NT、Unix、Oracle、Sybase、SQL Server 和其他应用包。通过使用 QALoad 与 EcoTools 可以在系统中生成一个负载，同时监控资源的利用问题。

4. 性能优化模块

性能优化模块 EcoScope 是一套定位于应用（即服务提供者本身）及其所依赖的所有网络计算资源的解决方案。EcoScope 可以提供应用视图，并标出应用是如何与基础架构相关联的。EcoScope 使用综合软件探测技术无干扰地监控网络，可自动发现应用、跟踪在 LAN/WAN 上的应用流量，采集详细的性能指标。EcoScope 将这些信息关联到一个交互式用户界面（Interactive Viewer）中，自动识别低性能的应用、受影响的服务器与用户、性能低下的程度，能自动调整应用和定位基础架构上的缺陷。

EcoScope 性能评分卡能很容易地显示出关键应用每时每刻是如何运行的，以及它们是否达到了预期的服务水平。对于必须满足服务水平协议的应用而言，EcoScope 能为之设置性能要求，并监控是否有偏离。如果一个应用超出了性能的上下限，EcoScope 将认为服务水平异常，并根据受影响用户的数量和性能降低的时间长短细分问题的严重程度。这些信息使 IT 维护人员能优先关注对业务影响最大的应用问题。

EcoScope 的性能评分卡以图形方式，按时间周期显示响应时间、流量、受应用影响的关键服务器和最终用户。在性能评分卡中，能通过比较和关联这些信息，确定应用使用量、响应时间、特定的最终用户和服务器之间的因果关系。在业务被阻碍前，跟踪每天的变化趋势，控制性能波动，快速找出性能瓶颈。

一旦 EcoScope 发现性能低下的应用，它将提供详细信息来隔离造成瓶颈的来源。EcoScope 图形化界面交互地观察单个受影响的工作站、服务器及网段。EcoScope 提供的大量信息有助于进行问题根源的分析、确定问题扩散的原因、受影响的服务器和用户、其性能受损是否有共性等。

EcoScope 对瓶颈的分析不限于网络基础架构和资源，而且包括其他关键计算资源，如桌面和服务器。

5. 测试数据自动生成模块

测试数据自动生成模块 TESTBytes 是一个用于自动生成测试数据的工具，为测试应用程序对数据库的访问自动生成测试数据。通过简单的点击式操作，就可以确定需要生成的测试数据的数据类型（包括特殊字符的定制），并通过与数据库的连接来自动生成数百万行的正确的测试数据，从而提高数据库开发人员、测试人员、数据仓库开发人员、应用开发人员的工作效率。

4.3.2 WinRunner 测试工具

WinRunner 是一种用于检验应用程序能否如期运行的企业级软件功能测试工具。通过自动捕获、检测和模拟用户交互操作，WinRunner 能识别出绝大多数的软件功能缺陷，从而确保那些跨越了多个功能点和数据库的应用程序在发布时尽量不出现功能性故障。WinRunner 的特点如下：

- 与传统的手工测试相比，它能快速、批量地完成功能点测试。
- 能针对相同的测试脚本执行相同的动作，从而消除人工测试所带来的理解上的误差。
- 能重复执行相同的动作，测试工作中最枯燥的部分可交由机器完成。
- 支持程序风格的测试脚本，一个高素质的测试工程师能借助它完成流程极为复杂的测试，通过使用通配符、宏、条件语句、循环语句等，还能较好地完成测试脚本的重用。
- 它针对于大多数编程语言和 Windows 技术，提供了较好的集成、支持环境，这对基于 Windows 平台的应用程序实施功能测试而言带来了极大便利。

WinRunner 的工作流程大致可以分为以下步骤。

1. 识别应用程序的 GUI

在 WinRunner 中可以使用 GUI Spy 来识别各种 GUI 对象，识别后 WinRunner 会将其存储到 GUI Map File 中。它提供两种 GUI Map File 模式：

- Global GUI Map File。
- GUI Map File per Test。

2. 建立测试脚本

在建立测试脚本时，一般先进行录制，然后在录制形成的脚本中手工加入需要的 TSL（与 C 语言类似的测试脚本语言），录制脚本提供两种模式：

- Context Sensitive。
- Analog。

3. 对测试脚本除错

在 WinRunner 中专门有一个 Debug Toolbar 用于测试脚本除错。可以使用 step、pause、breakpoint 等来控制和跟踪测试脚本和查看各种变量值。

4. 在新版应用程序中执行测试脚本

当应用程序有新版本发布时会对应用程序的各种功能包括新增功能进行测试。

5. 分析测试结果

分析测试结果在整个测试过程中最重要，通过分析可以发现应用程序的各种功能性缺陷。当运行完某个测试脚本后，会产生一个测试报告，从这个测试报告中能发现应用程序的功能性缺陷，能看到实际结果和期望结果之间的差异，以及在测试过程中产生的各类对话框等。

6. 回报缺陷

在分析完测试报告后，按照测试流程要回报应用程序的各种缺陷，然后将这些缺陷发给指定

人，以便进行修改和维护。

4.4 黑盒测试的操作步骤

为了保证测试工作科学、精确、全面、有序地进行，应该采取一边开发、一边测试的策略，使得开发工作与测试工作平行进行。黑盒测试的操作步骤应由以下 4 个阶段组成。

1. 测试计划

根据用户需求报告中关于功能要求和性能指标的规格说明书，定义相应的测试需求报告，选择测试内容，合理安排测试人员、测试时间及测试资源等。

2. 测试设计

将测试计划阶段制订的测试需求分解、细化为若干个可执行的测试过程，并为每个测试过程选择适当的测试用例。

3. 测试

建立测试过程，测试一般由单元测试、集成测试、系统测试等步骤组成，测试人员应对所发现的缺陷进行跟踪管理。

4. 测试评估

结合测试报告，对应用软件的质量和工作进度进行综合评价。

第5章　软件测试模型和测试工作指南

软件测试模型是对测试过程的一种抽象，用于定义软件测试的流程和方法，是确保软件工程质量的重要手段。软件测试工作指南是指导测试人员如何开展软件测试。本章重点讨论以下内容：

- 软件测试工作概述。
- 软件测试模型。
- 软件测试工作指南。

5.1　软件测试工作概述

5.1.1　软件测试工作流程

软件测试工作流程如图 5-1 所示。

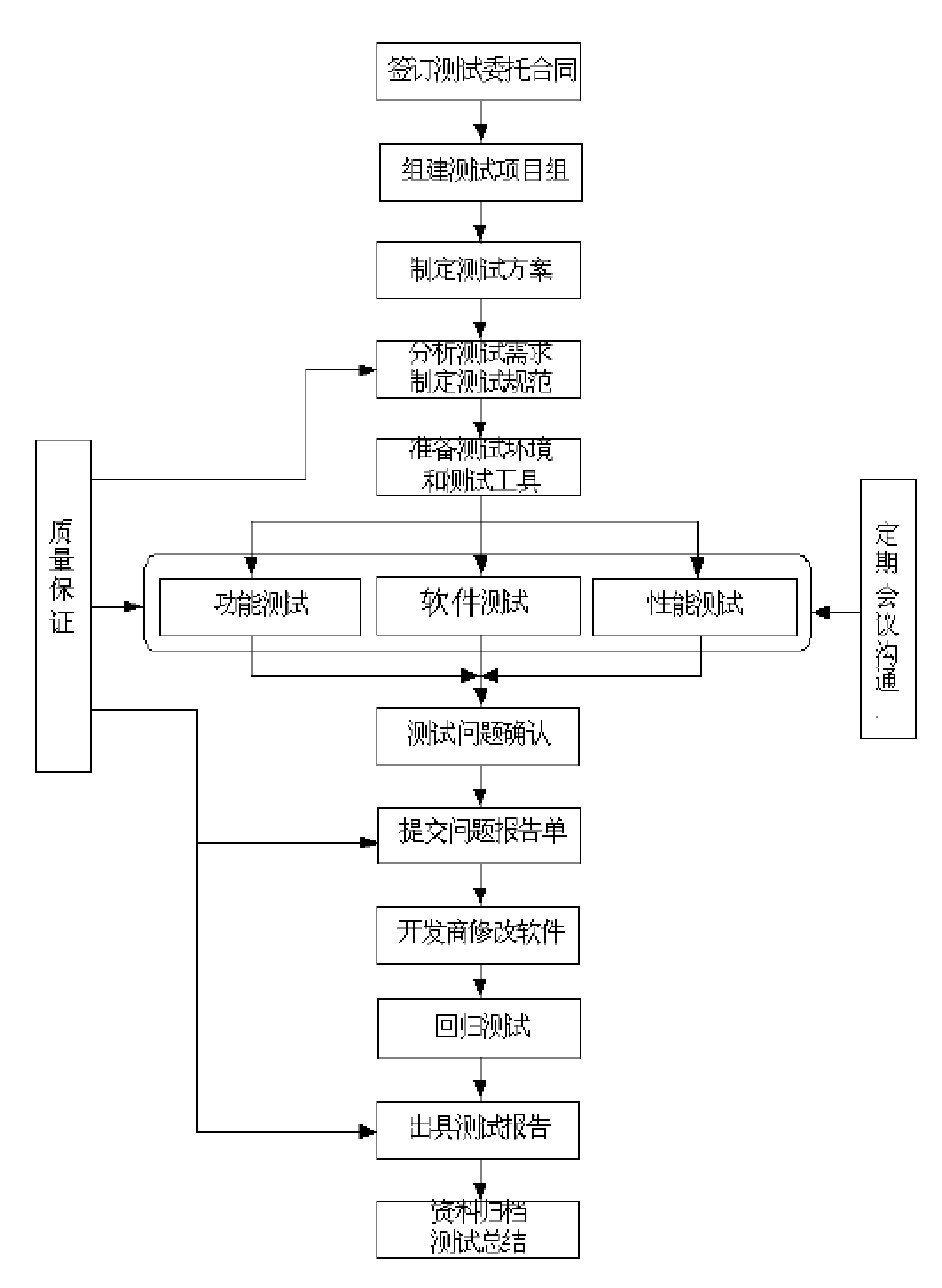


图 5-1　软件测试工作流程图

软件测试工作过程的详细流程如图 5-2 所示。

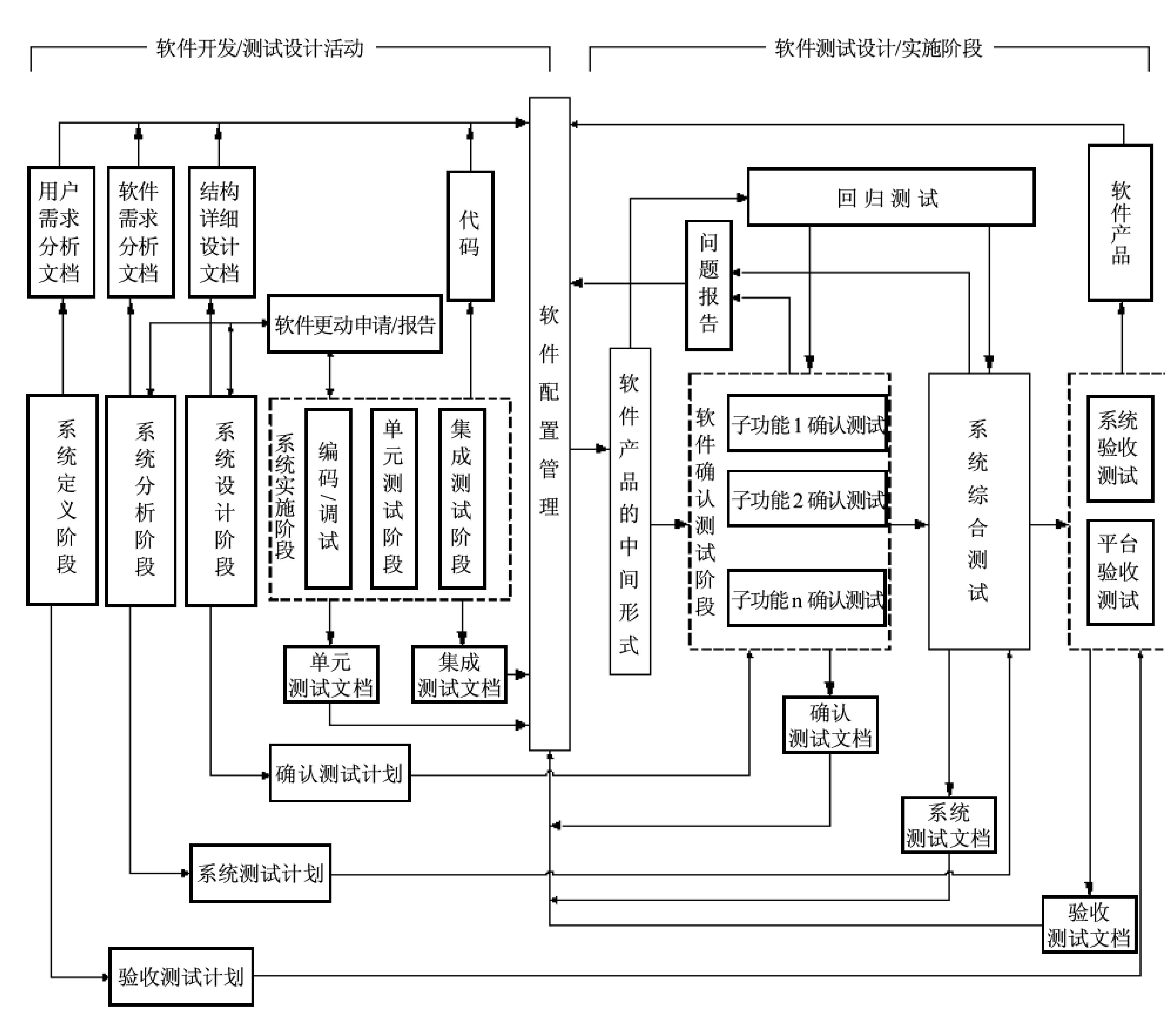


图 5-2　软件测试工作过程的详细流程

5.1.2　软件测试阶段

图 5-2 显示了软件系统的测试工作流程。从测试实际的前后过程来看，软件测试是由一系列的不同测试阶段组成的，这些软件测试的步骤分为需求分析、设计、单元测试、集成测试、功能验证、系统测试、验收测试、回归测试（维护）等，如表 5-1 所示。

表 5-1　软件测试阶段

阶段	输入和要求	输出
需求分析阶段	市场/产品需求定义、分析文档和相关技术文档。要求：需求定义要准确、完整和一致，真正理解客户的需求	需求定义中的问题列表、批准的需求分析文档、测试计划书的起草
设计阶段	产品规格设计说明、系统架构和技术设计文档、测试计划和测试用例。要求：系统结构的合理性、处理过程的正确性、数据库的规范化、模块的独立性、测试用例的有效性和完备性等，并清楚定义测试计划的策略、范围、资源和风险	设计问题列表、批准的各类设计文档、系统和功能的测试计划和测试用例、测试环境的准备

（续表）

阶段	输入和要求	输出
单元测试阶段	源程序、编程规范、产品规格设计说明书和详细的程序设计文档，要求：遵守规范、模块的内聚性、功能实现的一致性和正确性	缺陷报告、跟踪报告、完善的测试用例、测试计划，对系统功能及其实现等了解清楚；获得可组装的单元
集成测试阶段	通过单元测试的模块或组件、编程规范、集成测试规格来说明和程序设计文档、系统设计文档，要求：接口定义清楚且正确、模块或组件一起工作正常、能集成为完整的系统	缺陷报告、跟踪报告；完善的测试用例、测试计划；集成测试分析报告；集成后的系统
功能验证阶段	代码软件包（含文档）、功能详细设计说明书、测试计划和用例，要求：模块集成功能的正确性、适用性	缺陷报告、代码完成状态报告、功能验证测试报告
系统测试阶段	修改后的软件包、测试环境、系统测试用例和测试计划，要求：系统能正常、有效地运行，包括性能、可靠性、安全性、兼容性等	缺陷报告、系统性能分析报告、缺陷状态报告、阶段性测试报告
验收测试阶段	产品规格设计说明、预发布的软件包、确认测试用例，要求：向用户表明系统能够按照预定要求那样工作，使系统最终可以正式发布或向用户提供服务。用户要参与验收测试	用户验收报告、缺陷报告审查、版本审查、最终测试报告
维护	变更的需求、修改的软件包、测试用例和计划，要求：新的或增强的功能正常、原有的功能正常，不能出现回归缺陷	缺陷报告、更改跟踪报告、测试报告

这部分内容将在后面陆续进行讨论，这里不再赘述。

5.2 软件测试模型

软件测试模型主要有几种，如 V 模型、W 模型、H 模型、X 模型等，下面将一一进行介绍。

5.2.1 V 模型

V 模型是在快速应用开发（Rap Application Development，RAD）模型基础上演变而来，由于将整个开发过程构造成一个 V 字形而得名，如图 5-3 所示。V 模型强调软件开发的协作和速度，主要反映测试活动与分析设计关系，将软件实现和验证有机地结合起来，在保证较高的软件质量情况下缩短开发周期。

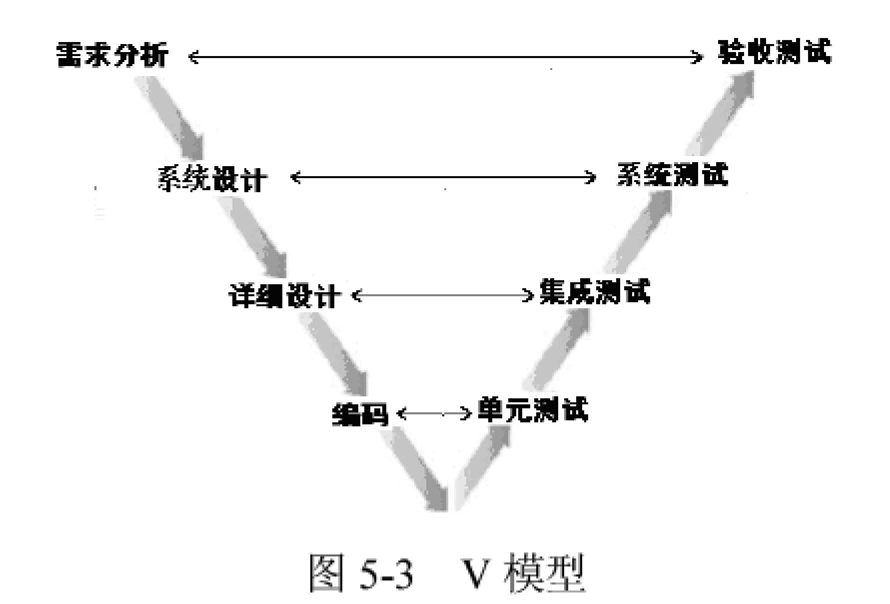


图 5-3 V 模型

在图 5-3 中，从水平对应关系看，左边是设计和分析，右边是对左边的测试。

- 需求分析对应验收测试：说明在做需求分析的同时，测试人员就可以阅读、审查需求分析的结果，从而了解产品的设计特性、用户的真正需求，确定测试目标，准备测试用例并策划测试活动。
- 系统设计对应系统测试：说明在做系统设计的同时，测试人员可以了解系统是如何实现的，设计系统的测试方案和测试计划，并事先准备系统的测试环境。
- 详细设计对应集成测试：说明在做详细设计的同时，测试人员可以参与设计，对详细设计进行评审并设计测试用例。
- 编码对应单元测试：说明在编码的同时设计测试用例，进行单元测试，尽快找出程序中的错误。

V 模型的价值在于它很明确地表明了测试过程中存在的不同级别，并且清楚地描述了这些测试阶段和研发过程期间各阶段的对应关系。V 模型适合企业级的软件开发，它清楚地揭示了软件开发过程的特性及其本质：

- 单元和集成测试应检测程序的执行是否满足软件设计的要求。
- 系统测试应检测系统功能、性能的质量特性是否达到系统要求的指标。
- 验收测试确定软件的实现是否满足用户需要或合同的要求。

但 V 模型存在一定的局限性，它仅仅把测试作为在编码之后的一个阶段，是针对程序进行的寻找错误的活动，而忽视了测试活动对需求分析、系统设计等活动的验证和确认的功能。

5.2.2　W 模型

W 模型由两个 V 模型组成，分别代表测试与开发过程，如图 5-4 所示。W 模型把开发活动看成是从需求到编码结束的一个串行过程，只有完成上一阶段活动后，才能进行下一阶段的活动，不支持迭代、自发性的变更调整。

W 模型强调测试伴随着整个软件开发周期，测试与开发同步进行，有利于尽早地发现问题的局限性，测试对象不仅仅是程序，需求和设计等同样需要进行测试。

W 模型有利于及时了解项目难度和测试风险，及早制定应对措施，这将显著减少总体测试时间、加快项目进度。

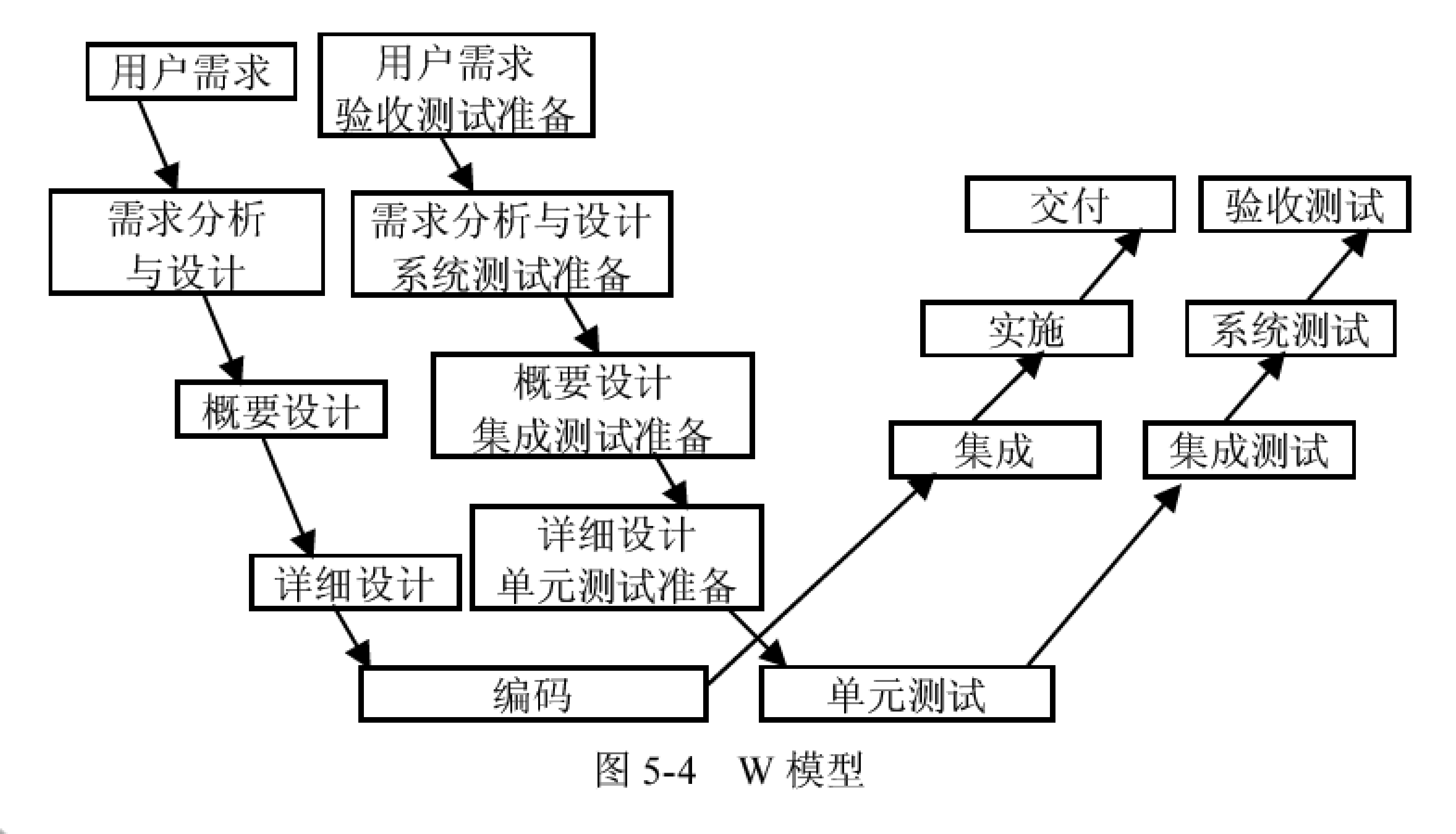


图 5-4　W 模型

但 W 模型也存在局限性，在 W 模型中需求分析、设计、编码等活动被视为串行的，同时，测试和开发活动也保持着一种线性的前后关系，上一阶段完全结束，才可正式开始下一个阶段的工作。

5.2.3　H 模型

在 H 模型中，软件测试的过程活动完全独立，形成了一个完全独立的流程，贯穿于整个产品的周期，与其他流程并发进行，某个测试点准备就绪后就可以从测试准备阶段进行到测试执行阶段；软件测试可以根据被测产品的不同分层进行，如图 5-5 所示。

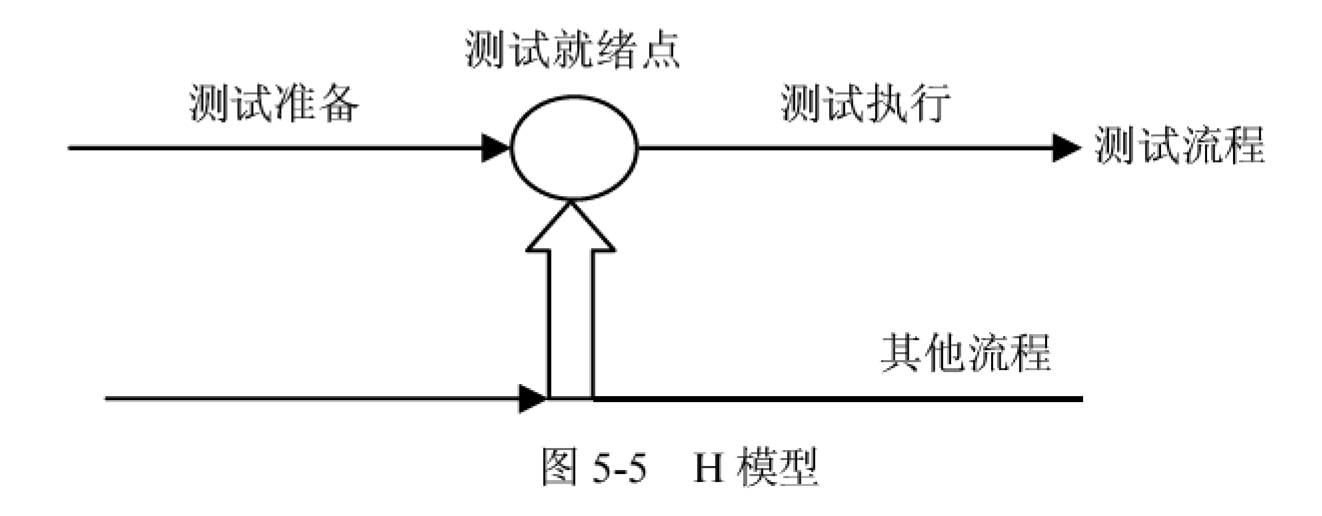


图 5-5　H 模型

5.2.4　X 模型

X 模型提出针对单独的程序片段进行相互分离的编码和测试，此后通过频繁的交接、集成、综合成为可执行的程序，如图 5-6 所示。

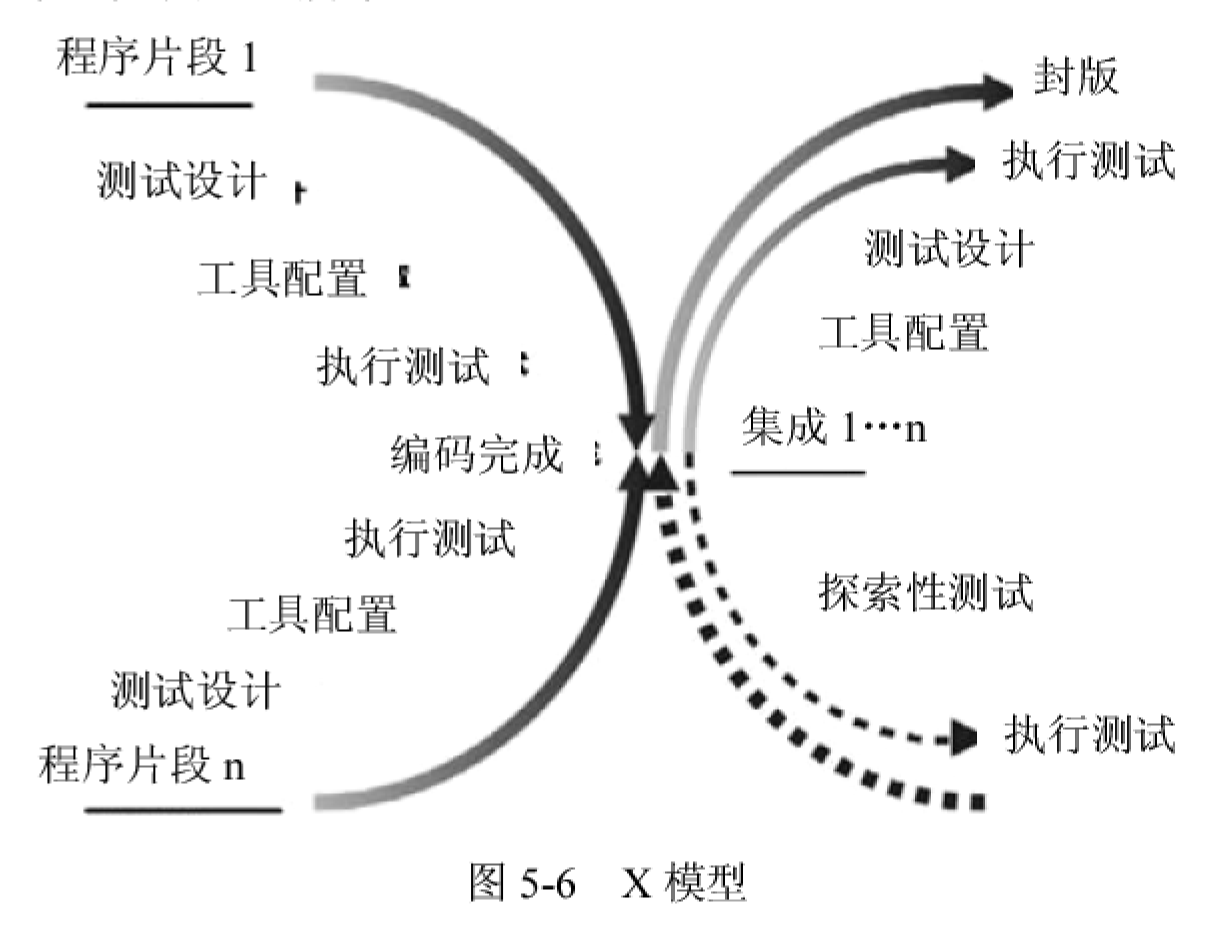


图 5-6　X 模型

5.3　软件测试工作指南

软件测试工作指南用于说明测试工作的目标，以及如何实施测试的方法和技术。

软件测试工作指南如图 5-7 所示。

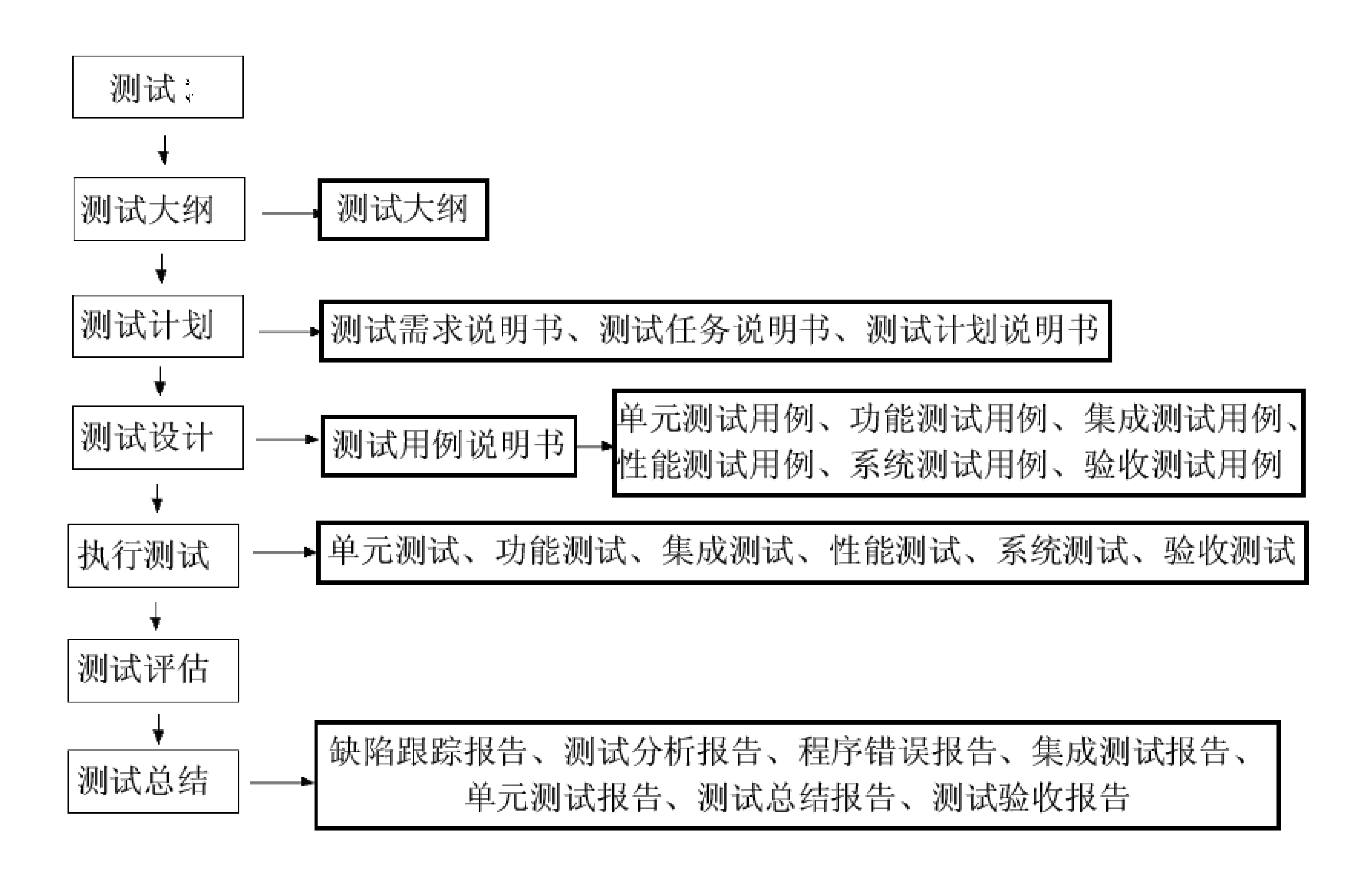


图 5-7　软件测试工作指南

图 5-7 中所涉及到的内容将在后面章节陆续讨论，这里不再赘述。

第6章　单元测试技术

单元测试又称为模块测试（程序测试），即集中力量来检验软件设计的最小单位——模块，模块的内聚程度高，每一个模块只完成一种功能，因此模块测试的程序规模较小、易于查错、发现错误后容易确定错误的位置。单元测试的目的在于发现各模块内部可能存在的各种差错。单元测试需要从程序的内部结构出发设计测试用例。多个模块可以平行地独立进行单元测试。

本章将重点讨论以下内容：

- 单元测试的内容。
- 单元测试的要点剖析。

6.1　单元测试的内容

模块是由程序员自己来完成的，程序员有责任编写功能代码，同时也就有责任为自己的代码进行单元测试。程序员交付的代码一定是通过编译的代码，但代码通过编译只能说明它的语法正确，却无法保证它的语义也一定正确，没有任何人可以轻易承诺这段代码的行为一定是正确的。

执行单元测试就是为了证明模块代码的行为和系统期望是一致的。这部分的测试工作是由程序开发人员（程序员）进行的。测试人员、QA（Quality Assurance）人员对单元测试工作的要求是：对所有局部和全局的数据结构、外部接口、程序代码的关键部分进行桌前检查和严格的代码审查。

单元测试是以程序设计说明书为指导，测试模块范围内的重要控制路径，以揭露错误。

当程序编写好以后将它录制在媒体上，或者直接由终端键盘输入到机器中进行调试。测试的相对复杂性和所发现的错误受到单元测试所限定的范围限制。它在执行过程中紧密依照程序框架对模块进行测试（调试），测试包含入口和出口的参数、输入和输出信息、错误处理信息、部分边界数值测试等，即需要在 6 个方面对所测模块进行检查。

1. 模块接口测试

模块接口测试是单元测试的基础，当模块通过外部设备进行输入/输出操作时，只有在数据能正确流入、流出模块的前提下，模块才能完成它的功能。

模块接口测试应考虑下列因素：

- 调用其他模块时所给的输入参数与模块的形式参数在个数、属性、顺序上是否匹配。
- 调用其他模块时所给实际参数的个数是否与被调模块的形参个数相同。
- 调用其他模块时所给实际参数的属性是否与被调模块的形参属性匹配。
- 调用预定义函数时所用参数的个数、属性和次序是否正确。

- 输入的实际参数与形式参数的个数是否相同。
- 输入的实际参数与形式参数的属性是否匹配。
- 输入的实际参数与形式参数的量纲是否一致。
- 是否修改了只做输入用的形式参数。
- 是否存在与当前入口点无关的参数引用。
- 是否修改了只读型参数。
- 对全程变量的定义，各模块是否一致。
- 是否把某些约束作为参数传递。
- 输出给标准函数的参数在个数、属性、顺序上是否正确。
- 限制是否通过形式参数来传送。
- 文件属性是否正确。
- OPEN/CLOSE 语句是否正确。
- 格式说明与输入输出语句是否匹配。
- 缓冲区大小与记录长度是否匹配。
- 文件使用前是否已经打开。
- 是否处理了输入/输出错误。
- 输出信息中是否有文字性错误。
- 在结束文件处理时是否关闭了文件。

2. 局部数据结构测试

局部数据结构是为了保证临时存储在模块内的数据在程序执行过程中完整、正确的基础。模块的局部数据结构往往是错误的根源，局部数据结构测试力求发现最常见的如下几类错误：

- 不合适或不相容的类型说明。
- 变量无初值。
- 变量初始化或缺省值有错。
- 不正确的变量名（拼错或不正确地截断）。
- 出现上溢、下溢和地址异常。

3. 路径测试

应对模块中重要的执行路径进行测试，错误的计算、不正确的比较或不正常的控制流都将导致执行路径的错误。路径错误应考虑下列因素：

- 运算的优先次序不正确或误解了运算的优先次序。
- 运算的方式错，即运算的对象彼此在类型上不相容。
- 算法错。
- 初始化不正确。
- 由于浮点数运算精度问题而造成的两值比较不等。
- 关系表达式中不正确的变量和比较符号表示不正确。
- 不正确地多循环一次或少循环一次。
- 错误的或不可能的循环终止条件。

- 当遇到发散的迭代时不能终止的循环。
- 不适当地修改了循环变量。

4. 边界条件测试

边界条件测试是单元测试中最重要的一项任务。边界条件测试是一项基础测试，也是后面系统测试中的功能测试的重点，边界测试执行得较好，可以大大提高程序的健壮性。边界条件测试应考虑下列因素：

- 程序内有一个 n 次循环，n 次循环应是 1~n，而不是 0~n。
- 由小于、小于等于、等于、大于、大于等于、不等于确定的比较值出错。
- 出现上溢、下溢和地址异常问题。

5. 错误处理测试

比较完善的模块设计要求能预见出错的条件，并设置适当的出错处理，以便在程序出错时，能对出错程序重新进行安排，从而保证其逻辑上的正确性。这种出错处理也应当是模块功能的一部分。错误处理测试应考虑下列因素：

- 出错的描述难以理解。
- 出错的描述不足以对错误定位，不足以确定出错的原因。
- 显示的错误与实际的错误不符。
- 对错误条件的处理不正确。
- 异常处理不当。

6. 代码书写规范测试

检查代码书写规范时应考虑下列因素：

- 模块设计程序框架流程图。
- 代码书写规范，对齐方式。
- 代码的注释。
- 参数类型、数据长度、指针、数组长度、大小。
- 输入/输出参数和结果。

由以上内容可以看出，单元测试具有以下优点。

- 单元测试是一种验证行为：程序中的每一项功能都是由测试来验证它的正确性。
- 单元测试是一种设计技术：单元测试迫使程序员将程序设计成易于调用和可测试的，是一种设计技术。
- 单元测试是一种编写文档的行为：单元测试是一种文档，它是展示函数或类如何使用的最佳文档。这份文档是可编译、可运行的，并且它能保持最新，永远与代码同步。
- 单元测试具有回归性：单元测试避免了代码出现回归，编写完成后可以随时随地的快速运行测试。
- 单元测试具有保证性：单元测试能够保证代码质量、代码的可维护性和可扩展性。

6.2 单元测试的要点剖析

单元测试是对每个程序的单体调试，其主要步骤如下。

01 程序语法检查。
02 程序逻辑检查。

在程序的逻辑检查之前，首先需要制作大量的测试数据（包括正常的数据、不同的数据、错误的数据），即假设一些输入数据和文件数据。测试数据直接影响了程序的调试工作，所以制作的数据应该满足以下几个条件：

- 数据应能满足设计上要求的上下限及循环次数。
- 数据应满足程序中的各种检验要求的错误数据。
- 数据应能适应于人工对程序的检查工作。

通过以上不同角度的数据检验，证明程序逻辑是对的，则程序的调试也就结束了。

在程序测试期，评价模块的 5 个主要特性如下：

- 模块接口。
- 局部数据结构。
- “重要”的执行路径。
- 错误处理路径。
- 影响上述几点的界限条件。

在其他任何测试开始之前，需要测试横穿模块接口的数据流。如果数据不是正确地进入和退出，其他的测试也就无从谈起。在程序测试中接口测试的清单如下：

- 输入参数的数目是否等于变元的数目。
- 参数与变元的属性是否匹配。
- 参数与变元的单位是否匹配。
- 传送给被调用模块的变元数是否等于参数的项目。
- 传送给被调用模块的变元属性是否同参数属性一致。
- 传送给被调用模块的变元单位是否同参数的单位一致。
- 属于内部的函数属性数目及变元次序是否正确。
- 对参数的任何访问是否与当前的入口点无关。
- 输入是否改动变元。
- 跨模块的全程量定义是否相容。
- 限制是否作为变元来传送。
- 参数是否被重复定义。

程序测试通常附属于编码步骤来考虑。在开发、复审了源代码并检查了语法正确性之后，考虑开发设计单元测试的情况。设计信息量复审为建立测试情况提供了指导，使得测试情况有可能发现上面讨论的各类错误。每个测试情况应有一组预期的结果。

由于模块不是一个独立的程序，必须为每个模块测试开发驱动软件和承接软件：

- 在大多数应用中，驱动软件和“主程序”并无区别。客观存在接受测试情况的数据，将这些数据送给模块，并打印有关的结果。
- 承接软件代替被测模块的下属模块。承接软件使用下属模块的接口，可以进行最少量的数据处理、打印入口检查信息，并将控制返回给它的上级模块。

驱动软件和承接软件代表开销，它们都是要书写的软件，但却不是与最后的软件产品一起支付的。当模块设计成高内聚时，单元测试就简化了。当一个模块只描述一个功能时，测试情况的数量就会减少，就可能会更容易地预计和发现错误。

程序测试实际上是为了发现错误而执行的程序，但测试不能发现所有的错误，即使是最彻底的切实可行的检测方法，也只能查出程序所存在的错误的一部分，其余的错误在实际使用过程中才能被逐渐发现。因为要检查每一种可能的情况，检测的数目是相当大的，实际上是不可能实现的；另一方面，一个软件的许多问题是无法通过输入来进行检查的，而是需要使用一些特殊的检测方法。

程序测试之后，还需要对每个程序书写一份程序测试说明书，以备系统今后修改维护。

程序测试说明书的主要内容如下：

- 说明程序测试数据制作的方法。
- 测试方法。
- 测试过程中所产生的问题。

其格式如表 6-1 所示。

表 6-1　测试说明书

程序名	功能名	调试日期	测试结果
数据制作方法：			
测试方法：			
测试问题： 单体程序测试问题 功能程序测试问题 子系统程序测试问题 系统测试问题			

第7章　功能测试技术

功能测试是对产品的功能进行验证，检查产品是否达到用户的功能要求。功能测试工作由程序员担当，测试的结果交系统设计人员、测试人员审核通过。本章重点讨论以下内容：

- 功能测试概述。
- 功能测试的流程。
- 功能测试用例的书写内容。

7.1　功能测试概述

1. 功能测试的基本概念

功能测试一般必须在完成单元测试后、集成测试前进行，而且是针对应用系统进行各功能测试。一般应用系统有多个功能（子系统），功能测试是基于产品功能说明书，是在已知产品所应具有的功能，从用户角度来进行的功能验证，以确认每个功能是否都能正常使用、是否实现了产品规格说明书的要求、是否能适当地接收输入数据而产生正确的输出结果等。功能测试包括用户界面测试、各种操作测试、不同的数据输入测试、逻辑思路测试、数据输出测试和存储测试等。对于功能测试，针对不同的应用系统，其测试内容的差异很大，但一般都可归为界面、数据、操作、逻辑、接口等几个方面。

功能测试的重点如下：

- 确认每个功能是否都能正常使用，每项功能是否符合实际要求。
- 是否实现了产品规格说明书的要求。
- 能否适当地接收输入数据而产生正确的输出结果。
- 系统的界面是否清晰、美观。
- 菜单、按钮操作是否正常、灵活，能处理一些异常操作。
- 是否能接收不同的数据输入（能接收正确的数据输入，对异常数据的输入可以进行提示、容错处理）。
- 数据的输出结果是否准确、格式清晰，能否保存和读取。
- 功能逻辑是否清楚，符合使用者习惯。
- 系统的各种状态按照业务流程而变化，并保持稳定。
- 支持各种应用的环境，能配合多种硬件周边设备，与外部应用系统的接口有效。
- 软件升级后，能否继续支持旧版本的数据。

2. 功能测试的工作

功能测试工作由程序员担当，测试的结果交系统设计、测试人员审核通过。

程序员进行功能测试时需要重点注意如下内容：

- 符合标准和规范。
- 直观性。
- 一致性。
- 灵活性。

3. 常用的功能测试方法

常用的功能测试方法如下。

- 页面链接检查：检查每一个链接是否都有对应的页面，并且页面之间切换正确。
- 相关性检查：检查删除/增加一项会不会对其他项产生影响，如果产生影响，这些影响是否都正确。
- 检查按钮的功能是否正确：检查如 update、cancel、delete、save 等功能是否正确。
- 字符串长度检查：检查输入字符串长度是否超出需求所说明的字符串长度的内容。
- 字符类型检查：检查在应该输入指定类型的内容的地方输入其他类型的内容（如在应该输入整型的地方输入其他字符类型）是否报错。
- 标点符号检查：检查输入内容包括各种标点符号，特别是空格、各种引号、回车键，检查系统处理是否正确。
- 中文字符处理：检查在可以输入中文的系统是否出现乱码或出错。
- 检查输出信息的完整性：在查看信息和 update 信息时，查看所填写的信息是不是全部输出，输出信息和添加信息是否一致。
- 信息重复：检查在一些需要命名且名字应该惟一的信息处输入重复的名字或 ID，查看系统有没有处理、是否报错，重名包括是否区分大小写以及在输入内容的前后输入空格等，系统是否作出正确处理。
- 检查删除功能：检查在一些可以一次删除多个信息的地方，不选择任何信息，单击 delete 键，查看系统如何处理、是否出错；然后选择一个和多个信息进行删除，查看是否能正确处理。
- 检查添加和修改是否一致：检查添加和修改信息的要求是否一致，例如添加要求必填的项，修改也应该必填；添加规定为整型的项，修改也必须为整型。
- 检查修改重名：检查修改时把不能重名的项改为已存在的内容，查看系统是否处理、报错。同时，也要注意会不会报重名的错。
- 重复提交表单：检查一条已经成功提交的记录，后退（back）后再提交，查看系统是否进行处理。
- 检查多次使用 back 键的情况：检查在有 back 的地方，back 回到原来页面，再 back，重复多次，查看是否会出错。
- search 检查：检查在有 search 功能的地方输入系统存在和不存在的内容，查看 search 结果是否正确，如果可以输入多个 search 条件，可以同时添加合理和不合理的条件，查看系统

处理是否正确。

- 输入信息位置：检查在光标停留的地方输入信息时，光标和所输入的信息是否会跳到其他地方。
- 上传下载文件检查：检查上传下载文件的功能是否能够实现、上传文件是否能够打开、对上传文件的格式有何规定、系统是否有解释信息等。
- 必填项检查：检查应该填写的项没有填写时系统是否都进行了处理、对必填项是否有提示信息，如在必填项前加“*”。
- 快捷键检查：检查是否支持常用快捷键，如 Ctrl+C、Ctrl+V、Backspace 等，对一些不允许输入信息的字段，如选人、选日期对快捷方式是否进行了限制。
- 回车键检查：检查在输入结束后直接按回车键时系统处理如何、是否会报错。

7.2 功能测试的流程

在单个程序测试成功的基础上进行功能测试。功能测试是综合的测试，因为若干个程序组成一个功能，所以功能测试是将功能内所有程序按处理流程图的次序串联起来进行的综合测试，如图 7-1 所示。

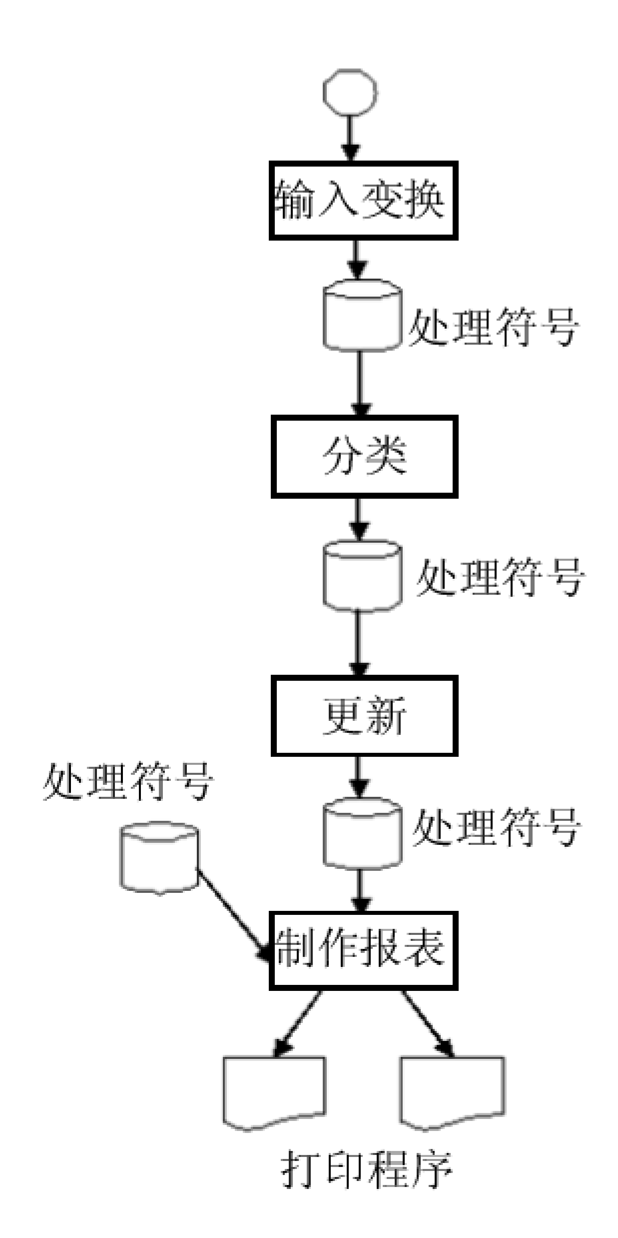


图 7-1 功能处理流程图

在图 7-1 中，功能处理流程图有 4 个处理符号，每个处理符号代表着一个程序，若计算机系统本身已具有服务程序，则输入变换和分类处理程序可以调用服务程序，因此这个功能需要编写更新和制表两个程序。所谓功能测试，即从处理流程输入开始，直至最后执行打印程序为止，如果没有逻辑问题就说明测试成功，此功能符合设计要求。

测试工作由程序员担当，测试的结果交由系统设计人员审核通过。

在各个功能测试成功的基础上，进行各个子系统的测试。每个子系统是由若干个功能组成的，子系统设计的成功与否，不仅决定于每个功能测试的成功与否，还决定了按信息传递先后次序串联起来的功能测试成功与否，因此，子系统的测试是一种连接的测试。

系统功能测试由于范围大、子系统（功能）多、信息多、联系广，所以测试之前需要做好充分的准备工作，如准备功能联系图、准备处理联系图、准备作业联系图。

1. 功能联系图

对于子系统范围内的所有功能，它们之间的相互关系要充分了解，把各功能通过接口文件相互联系起来，绘制成为联系图，如图 7-2 所示。

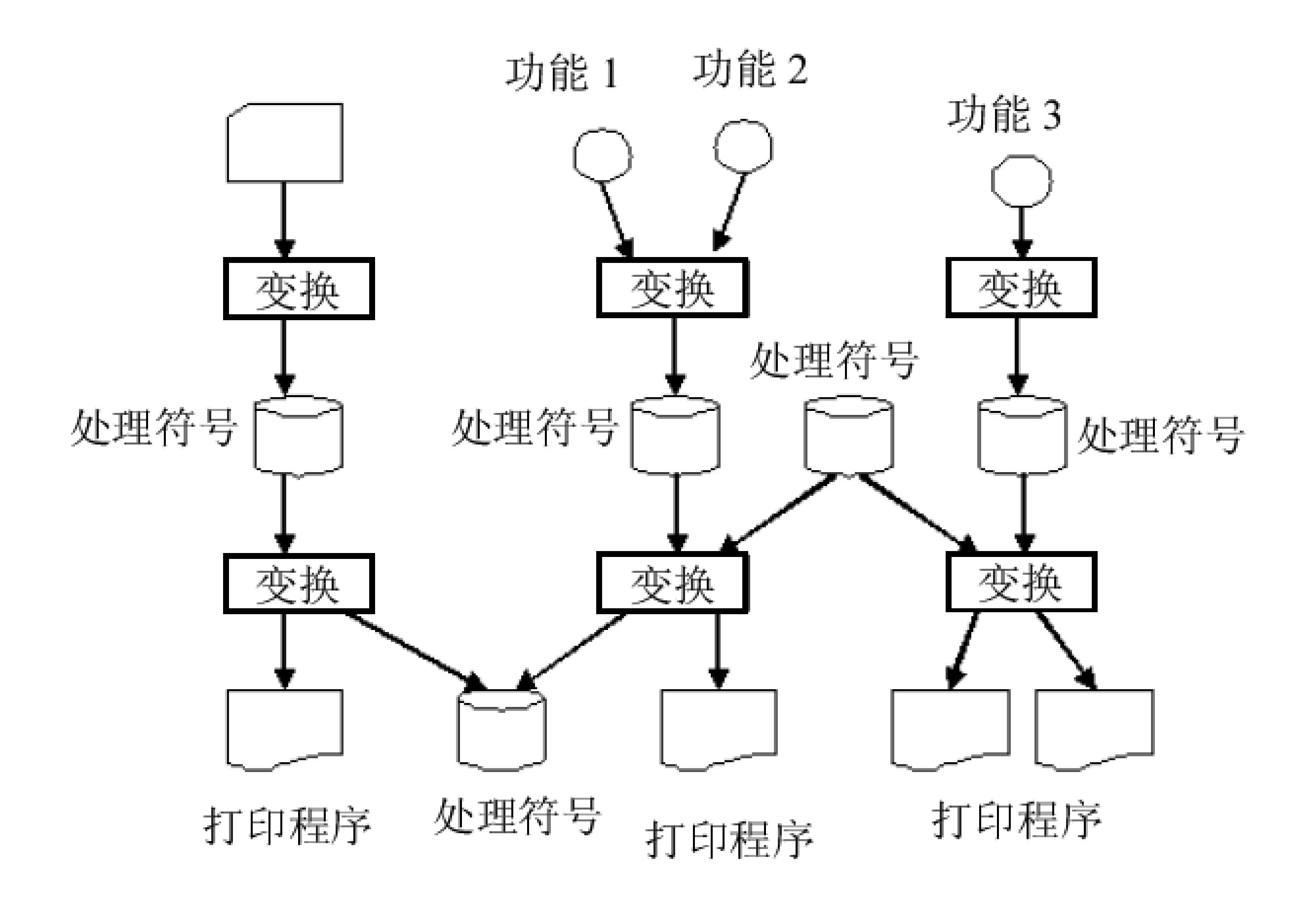


图 7-2　功能联系图

2. 处理联系图

对于子系统范围内的所有功能，它们的处理先后次序必须做好安排，以免处理时衔接不上。把各功能按处理上要求的时间先后绘制成处理联系图，如图 7-3 所示。

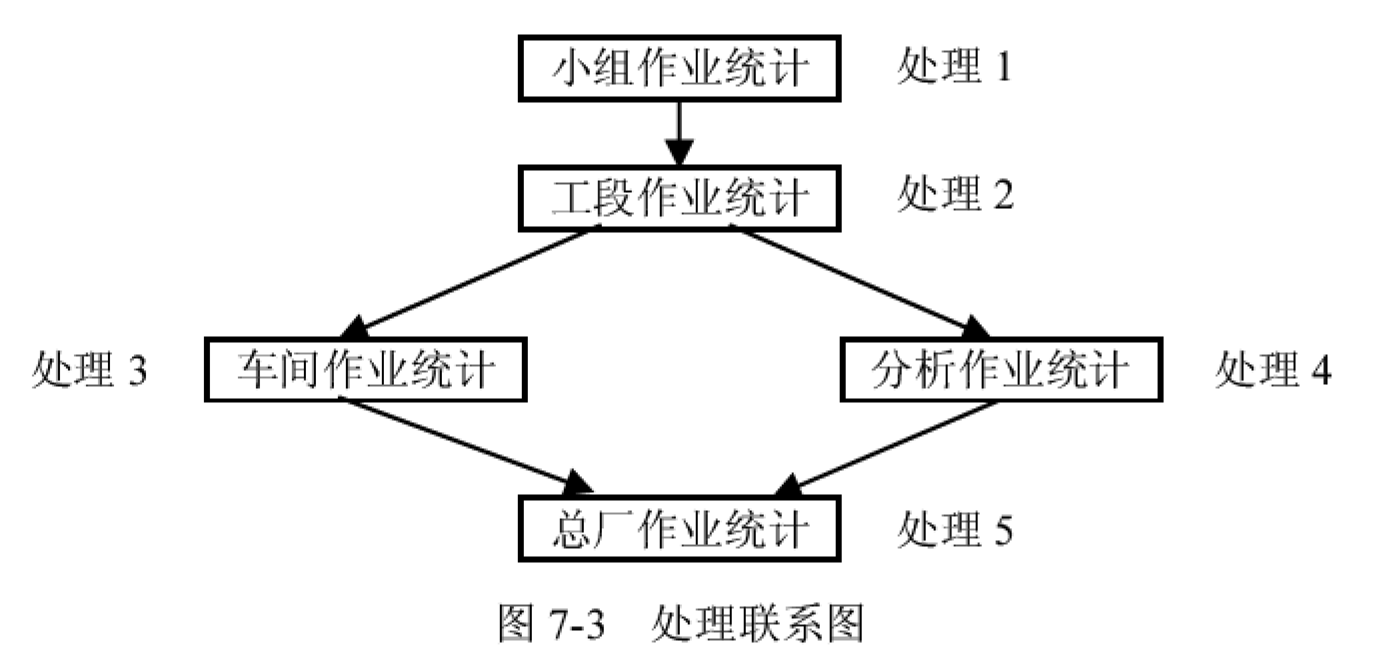


图 7-3　处理联系图

3. 作业联系图

子系统范围内的所有功能可以分成若干个作业，每个作业由若干个程序组成，作业是上机执

行单位，为了掌握作业执行的先后次序，要以作业为单位，相互联系起来绘制成作业联系图，如图7-4所示。

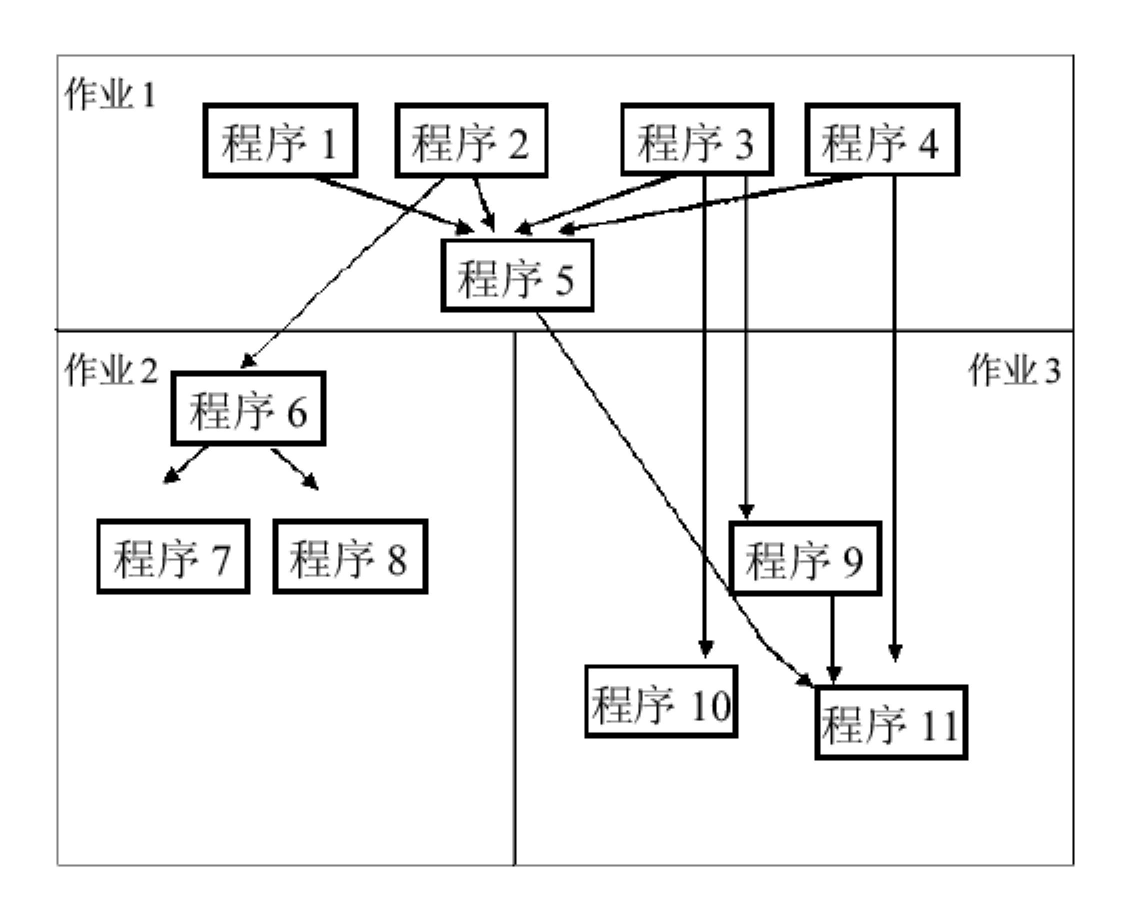


图 7-4　作业联系图

完成了以上准备工作以后，为了确保各功能、各程序的逻辑性，还需要再制作一套新的测试数据进行测试，以便进一步考验程序。对测试中发现的问题要及时地进行程序的修改，边测边改，直至测试成功。程序修改后，应该在测试说明书中说明测试中发现的问题、修改原因和修改内容，作为程序测试的补充说明。

在子系统（功能）的测试过程中，必须要合理地组织人员。将系统设计人员和程序设计人员统一调度使用，分成三个部分：一部分上机测试人员，一部分机下检查核对人员，还有一部分是程序修改人员。这三方面人员应该紧密配合、互相协调，从而保证子系统测试工作的顺利进行。

7.3　功能测试用例的书写内容

功能测试用例的书写内容主要有测试用例的需求、测试用例的组织方式、测试用例的表述要点等。

1. 测试用例的需求

测试用例的主要需求如下：

- 需求说明及相关文档。
- 相关的设计说明（概要设计、详细设计等）。
- 与开发组交流对需求理解的记录。
- 从项目文档、资料中分解出若干小的“功能点”，理解“功能点”，编写相应的测试用例。

2. 测试用例的组织方式

原则上程序员负责测试用例的书写，而系统设计人员负责进行审核。

3. 测试用例的表述要点

测试用例的表述要点，即用例中应当包含的信息，一个测试用例应该包含以下信息：

- 软件或项目的名称。
- 软件或项目的版本（内部版本号）。
- 功能模块名。
- 测试用例的简单描述，即该用例执行的目的或方法。
- 测试用例的参考信息。
- 用例的前置条件，即执行本用例必须要满足的条件，如对数据库的访问权限等。
- 用例的编号（ID）。
- 步骤号、操作步骤描述、测试数据描述。
- 预期结果和实际结果。
- 开发人员和测试人员的姓名。
- 测试执行日期。

第8章　网络测试和软件安装测试技术

网络是软件和用户应用系统基本的、不可缺少的环境，相当重要。网络测试涉及到网络产品测试和网络本身测试两方面。

软件安装测试分为共享软件安装测试和用户应用系统软件安装测试，是共享软件和用户应用系统软件在网络环境下高效可用的保证。

网络测试和软件安装测试为了保证共享软件和用户应用系统软件在网络环境下高效可用、达到设计要求，所采购产品在功能、性能和安全性等方面符合用户应用系统的需求。通过软件安装测试能够准确判断相关设备所实现的功能以及其自身的性能；通过网络测试，能够直接反应网络状况、检验应用系统的建设质量、定位和排查问题，为用户的正常应用提供保证。通过网络测试和软件安装测试有助于发现设计规范存在的问题，从而返回设计阶段重新设计或修正设计。

本章重点讨论以下内容：

- 网络产品的测试。
- 网络本身的测试。
- 软件安装的测试。

8.1　网络产品的测试

为有效判断项目是否能实现应用系统规格说明书中的要求，将根据网络建设方案及相关技术要求，对项目应用系统所涉及到的网络系统进行全面测试，具体将分为产品和网络两大部分进行测试，产品包括所涉及到的交换机、防火墙、服务器、安全产品等，在该范围内将对产品的不同型号进行测试，同型号产品进行抽样测试。网络测试包括在整个网络内不同区域间的性能、网络所实现的功能、各区域所采用的安全策略设置等进行测试。网络产品测试一般是第三方评测机构从最终用户的角度出发，按照专业的网络测试方法，模拟网络中的正常应用，进行全面的网络系统测试。

网络产品测试主要包括以下内容：

- 防火墙产品测试。
- 入侵检测产品测试。
- 入侵防护测试。
- 漏洞扫描测试。
- 防病毒测试。
- 交换机测试。
- 服务器测试。

网络产品测试项目的实施过程如图 8-1 所示。

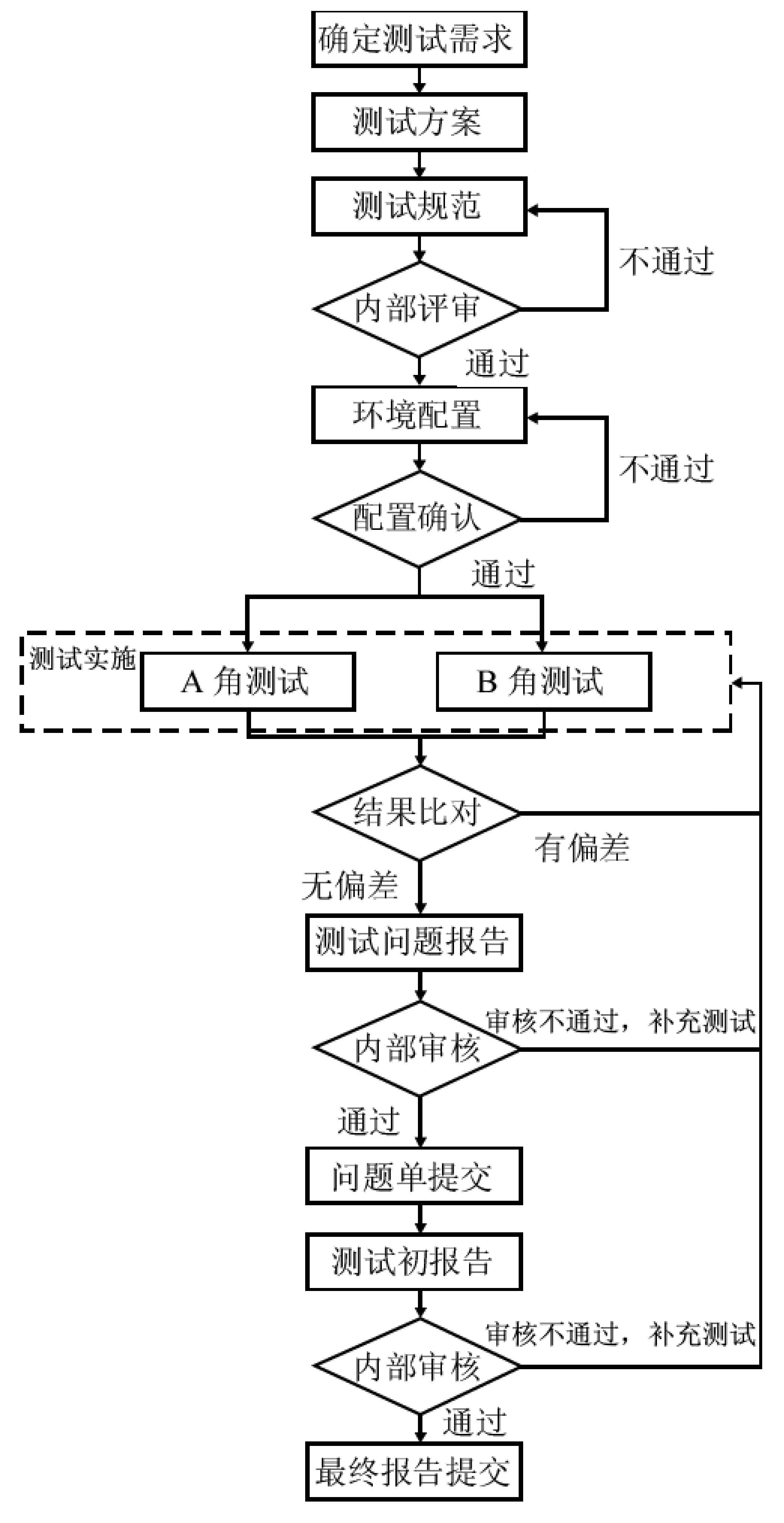
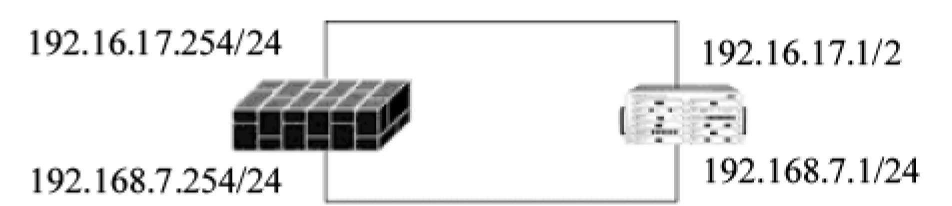


图 8-1　网络产品测试项目的实施过程

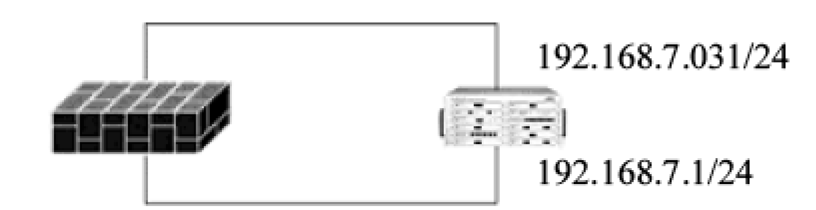
8.1.1　防火墙产品测试

产品测试是直接将测试设备与被测设备相连，从而测试其性能。防火墙产品测试如图 8-2 和图 8-3 所示。

防火墙性能测试拓扑图（一）吞吐量



路由模式（含 KAT）性能测试拓扑图



透明模式性能测试拓扑图

图 8-2　防火墙产品测试（a）

防火墙性能测试拓扑图（二）并发&背景流

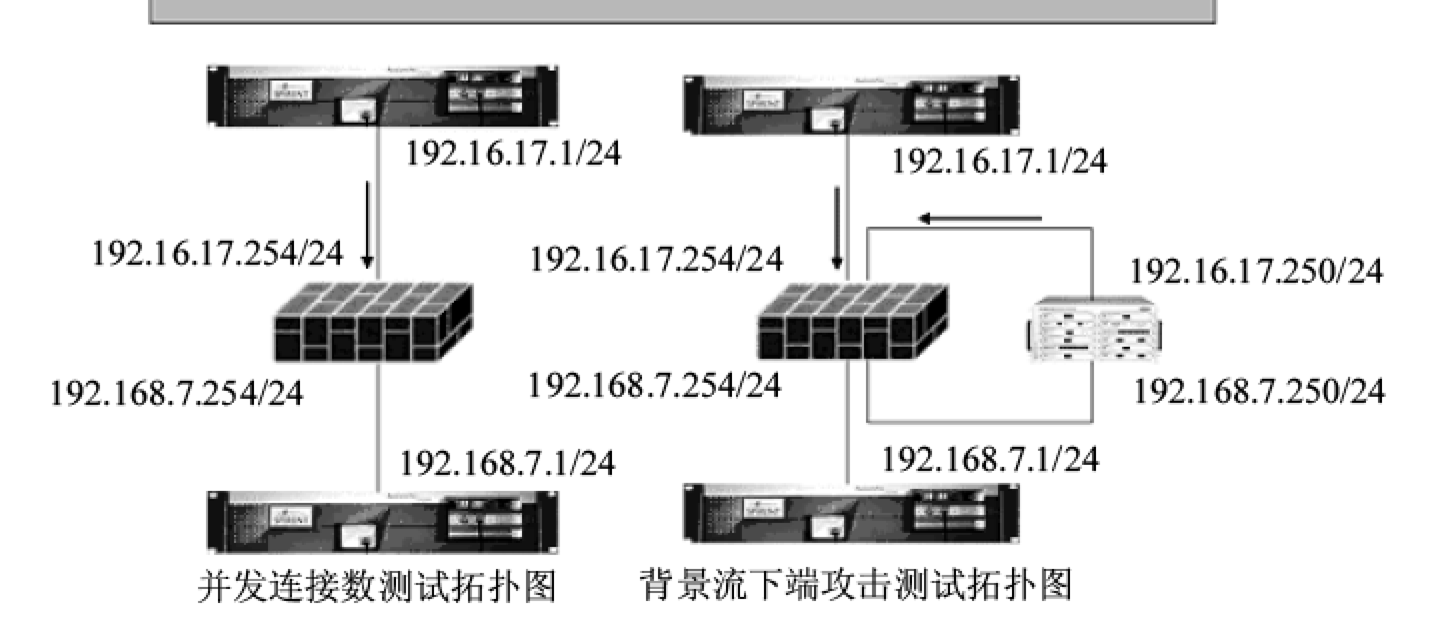


图 8-3　防火墙产品测试（b）

进行防火墙产品测试时需要重点注意如下内容。

1. 物理特性

测试物理特性时的重点注意事项如下：

- 硬件参数。
- 网络接口。
- 内部配置。

2. 基本功能

测试基本功能时的重点注意事项如下：

- 缺省配置。
- 工作模式。
- 包过滤。
- IP/MAC 地址绑定。

- 规则检查。
- 网络地址转换。
- 应用代理。
- 内容过滤。

3. 管理功能

测试管理功能时的重点注意事项如下：

- 管理方式。
- 管理分级。
- 管理认证。
- 通信加密。
- 安全措施。
- 集中管理。
- 日志分类。
- 日志分析。
- 日志管理。
- 状态监控。
- 系统升级。
- 其他功能。

4. 安全性

测试安全性时的重点注意事项如下：

- 稳定与可靠性。
- 抗攻击性。

5. 性能

测试性能时的重点注意事项如下：

- 吞吐。
- 延迟。
- 并发连接数。
- 每秒新建连接速率。
- HTTP 响应时间。
- 可用带宽。

8.1.2 入侵检测产品测试

入侵检测产品的测试拓扑图如图 8-4 和图 8-5 所示。

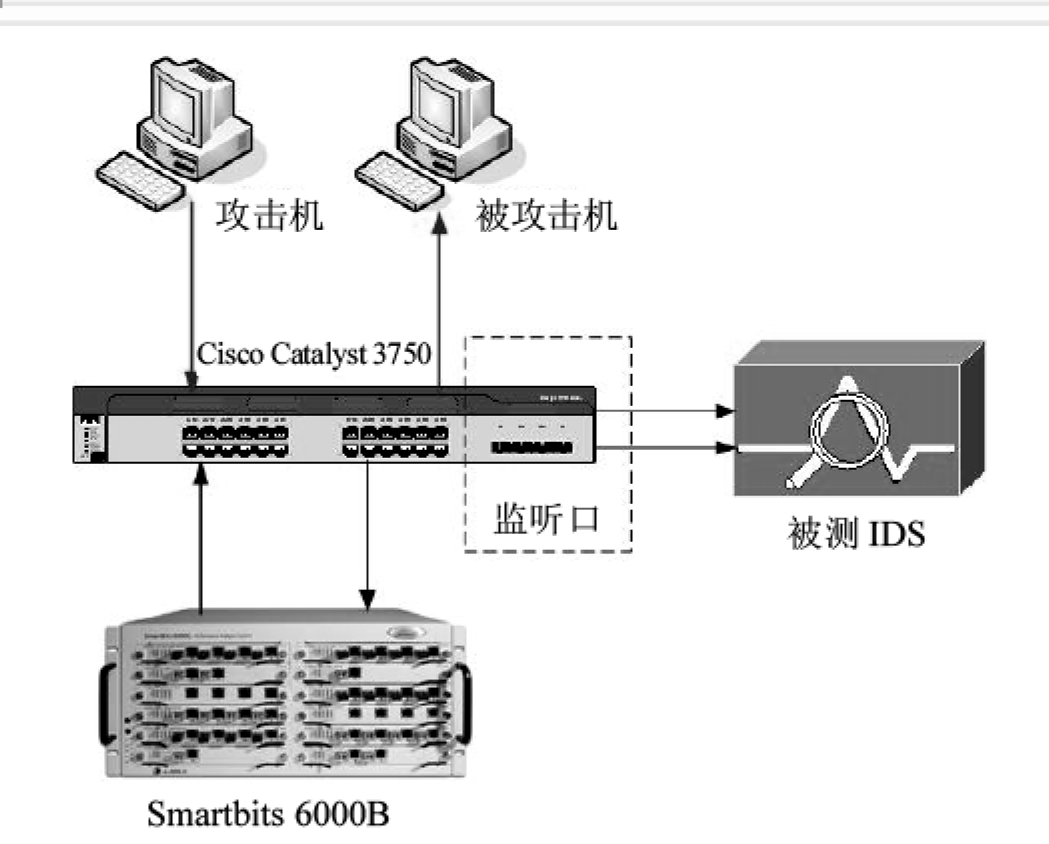


图 8-4 入侵检测产品测试拓扑图（a）

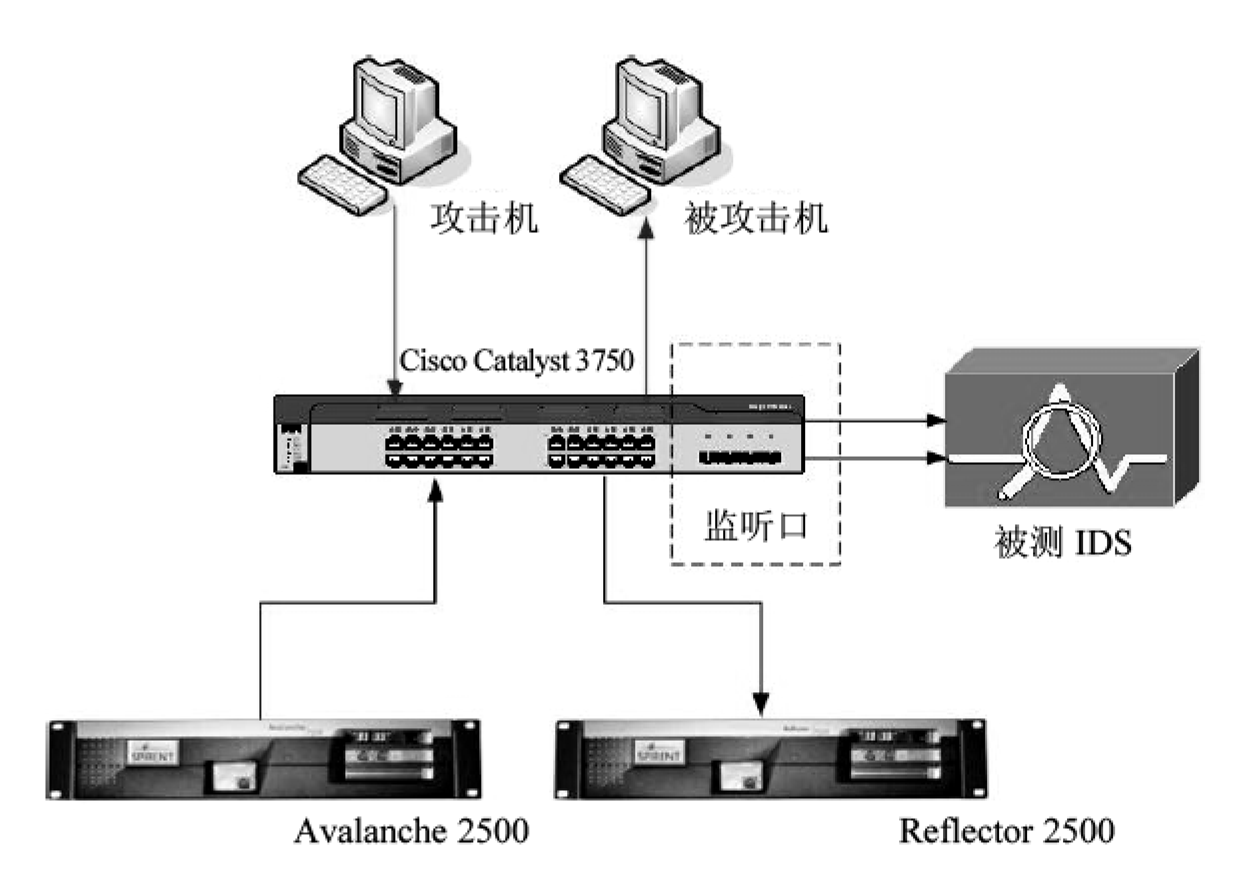


图 8-5 入侵检测产品测试拓扑图（b）

进行入侵检测产品测试时需要重点注意如下内容。

1. 物理特性

测试物理特性时的重点注意事项如下：

- 硬件参数。
- 网络接口。
- 内部配置。

2. 管理功能

测试管理功能时的重点注意事项如下：

- 集中管理。
- 跨网段管理。
- 管理形式。
- 管理难易程度。

- 自定义规则设置管理。
- 功能设置逻辑关系。
- 管理权限分级。
- 管理员鉴别失败处理。

3. 基本功能

测试基本功能时的重点注意事项如下：

- 记录并显示。
- 报警信息。
- 联动。
- 升级更新。
- 网络流量检测。
- 连接检测。
- 报警信息处理。
- 重点服务器检测。
- 报警信息过滤。
- 报警信息分级。
- 报表内容输出查询。
- 内容恢复。
- 事件回放。

4. 检测能力

测试检测能力时的重点注意事项如下：

- 网络扫描类。
- 非授权访问类。
- 后门木马类。
- 蠕虫类。
- 变形逃避类。

5. 检测性能

测试检测性能时的重点注意事项如下：

- TCP 背景压力测试。
- UDP 背景压力测试。
- HTTP 背景压力测试。

6. 自身安全性

测试自身安全性时的重点注意事项如下：

- 通信加密处理。
- 通信认证。

- 延迟报警。
- 健壮性测试。

8.1.3 入侵防护测试

进行入侵防护测试时需要重点注意如下内容。

1. 物理特性

测试物理特性时的重点注意事项如下：

- 产品外观。
- 内部配置。
- 网络适配器。
- 扩展接口。
- IPS 控制台检验测试，包括对控制台的硬件最低配置要求（CPU、内存、硬盘等）、对控制台的软件最低配置要求（操作系统、数据库、浏览器等）、对控制台的其他要求等。

2. 管理功能

测试管理功能时的重点注意事项如下：

- 集中管理。
- 远程管理。
- 管理形式。
- 自定义规则设置。
- 管理分级。
- 管理员鉴别失败处理。

3. 基本功能

测试基本功能时的重点注意事项如下：

- 记录并显示。
- 响应方式。
- 升级更新。
- 流量检测。
- 连接检测。
- 报警信息处理。
- 报警信息过滤。
- 报警信息分级。
- 报表内容输出查询。
- 状态检测。

4. 检测能力

测试检测能力时的重点注意事项如下：

- 网络扫描类。
- 非授权访问类。
- 后门木马类。
- 蠕虫类。
- 变形逃避类。

5. 自身安全性

测试自身安全性时的重点注意事项如下：

- 通信加密。
- 通信认证。
- 延迟报警。

6. 性能

测试性能时的重点注意事项如下：

- 吞吐量。
- 延迟。
- 丢包率。
- TCP 背景压力测试。
- UDP 背景压力测试。
- 并发连接数。
- 新建连接速率。
- 可用带宽（最大流量）。
- HTTP 并发背景压力测试。
- HTTP 流量背景压力测试。

8.1.4 漏洞扫描测试

进行漏洞扫描测试时需要重点注意如下内容。

1. 系统功能

测试系统功能时的重点注意事项如下：

- 部署和管理。
- 系统升级能力。
- 系统配置测试。
- 扫描策略定制。
- 信息收集能力。
- 资产管理。
- 扫描文档和报表。
- 报表信息完善程度和正确测试。

- 系统的日志、审计功能与系统的安全策略。
- 联动。
- 文档。

2. 漏洞扫描

测试漏洞时的重点注意事项如下：

- 误报和漏报。
- 系统的脆弱性，主要包括邮件服务的脆弱性、FTP 服务的脆弱性、Web 服务的脆弱性、DNS 服务的脆弱性、其他已知 TCP/IP 服务的脆弱性、SNMP 服务的脆弱性、口令的脆弱性、NFS 服务的脆弱性、路由器/交换机的脆弱性等。
- 漏洞的选择。
- 系统脆弱性的发现。
- 系统智能化程度。

3. 性能

测试性能时的重点注意事项如下：

- 扫描速度测试。
- 单位时间扫描的主机数。
- 单位时间正确探测到的漏洞数目 。
- 端口扫描速度和主机系统特征识别速度。
- 资源开销。

4. 可用性

测试可用性时的重点注意事项如下：

- 扫描的持续性。
- 对目标系统的影响。
- 自动化程度。
- 界面的友好程度。

8.1.5 防病毒测试

进行防病毒测试时需要重点注意如下内容。

1. 功能测试

功能测试包括 7 个方面，即安全性、信息反馈、增强功能、系统升级、监控定制、查杀定制、性能等。

（1）安全性

测试安全性时的重点注意事项如下：

- 安装前执行扫描。
- 安装后执行扫描。
- 启动时扫描。
- 密码保护。
- 手动隔离。

（2）信息反馈

测试信息反馈时的重点注意事项如下：

- 日志系统。
- 报警。
- 系统提示。

（3）增强功能

测试增强功能时的重点注意事项如下：

- 引导区备份/修复。
- 文件备份/修复。
- 扫描系统漏洞。
- 多语言支持。
- 文件指纹。
- 防火墙功能。

（4）系统升级

测试系统升级时的重点注意事项如下：

- 在线升级。
- 本地升级。
- 服务器升级。
- 配置升级模块。
- 调整升级间隔。
- 定制自动升级。

（5）监控定制

测试监控定制时的重点注意事项如下：

- 排除/选取。
- 处理方式。

（6）查杀定制

测试查杀定制时的重点注意事项如下：

- 排除/选取。
- 处理方式。

（7）性能

测试性能时的重点注意事项如下：

- CPU 占用。
- 内存占用。
- 磁盘空间占用。
- 海量文件扫描。

2. 查毒测试

进行查毒测试时的重点注意事项如下：

- 病毒样本测试。
- 变形病毒测试。
- 压缩格式测试。
- 压缩层数测试。
- 嵌套/混合压缩测试。
- 加壳测试。
- 杀毒测试。
- 病毒样本测试。
- 变形病毒测试。

除以上测试项目外，还需要进行压缩格式测试、压缩层数测试、嵌套/混合压缩测试、加壳测试、监控测试等。

8.1.6 交换机测试

进行交换机测试时需要重点注意如下内容。

1. 物理特性

测试物理特性时的重点注意事项如下：

- 外观测试。
- 端口配置。
- 电源配置。
- 模块配置。
- 指示灯、控制键的设置。

2. 基本功能

测试基本功能时的重点注意事项如下：

- 10Base-T 直通/交叉线自动协商。
- 10Base-T 半双工/全双工自动协商。
- 100Base-T 直通/交叉线自动协商。

- 100Base-T 半双工/全双工自动协商。
- 全双工线路的流量控制。

3. 管理功能

测试管理功能时的重点注意事项如下：

- 网络管理（配置管理）。
- 网络管理（安全管理）。
- 网络管理（端口管理）。
- 被测设备接口组对象的查询。
- 被测设备 Ethernet 状态的配置。

4. 业务功能

测试业务功能时的重点注意事项如下：

- 正常连接，交换数据帧。
- MAC 地址学习。
- 组播测试。
- 地址过滤功能测试。
- 查询软件版本信息。
- 闭塞用户侧端口。
- 恢复被闭塞的端口。
- 状态查询。
- 设备重起。

5. 性能

测试性能时的重点注意事项如下：

- 吞吐量。
- 突发长度。
- 突发间隔。
- 超负荷。
- 地址缓存能力。
- 交换机时延。
- 轻载。
- 重载时延。
- 交换机时延抖动。
- 轻载时延抖动。
- 重载时延抖动。
- 交换机丢包率。
- 轻载丢包率。
- 重载丢包率。

6. 协议

测试协议时的重点注意事项如下：

- VLAN 功能测试。
- 按端口划分 VLAN 验证。
- 按 MAC 划分 VLAN 验证。
- VLAN TRUNK 功能验证。
- 生成树协议测试。
- 产生生成树 1。
- 产生生成树 2。
- 重新产生生成树 1。
- 重新产生生成树 2。

8.1.7 服务器测试

进行服务器测试时需要重点注意如下内容。

1. 服务器的稳定性

服务器的稳定性是最重要的，如果在稳定性方面不能够保证业务运行的需要，再高的性能也是无用的。正规的服务器厂商都会对产品在不同温度和湿度下的运行稳定性进行测试。重点需要考虑的是冗余功能，如数据冗余、网卡冗余、电源冗余、风扇冗余等。

2. 服务器的压力测试

服务器压力测试是验证各事务在最大并发数下事务响应时间能否达到客户要求、系统各性能指标在这种压力下是否还在正常数值之内、系统是否会因这样的压力导致不良反应（如宕机、应用异常中止等）。

3. 服务器的可靠性测试

服务器的可靠性是由服务器的平均无故障时间 MTBF（Mean Time Between Failure）来度量的，故障时间越少，服务器的可靠性越高。

4. 服务器的可管理性测试

服务器的可管理性是 PC 服务器的标准性能。服务器管理有两个层次：硬件管理接口和管理软件。管理的内容可以包括性能管理、存储管理、可用性/故障管理、网络管理、安全管理、配置管理、软件分发、统计管理和技术支持管理等。

5. 服务器的可用性测试

关键的企业应用都追求高可用性服务器，系统 24×7 不停机、无故障运行。

6. 服务器的易用性测试

服务器应多采用国际标准，机箱设计应科学合理、拆卸方便，可热插拔部件较多，可随时更换故障部件，而且随机配有完善的用户手册，可以指导用户迅速简单的安装和使用。

7. 服务器的可扩展性测试

服务器的可扩展性是服务器的重要性能之一。服务器在工作中的升级特点，表现为工作站或用户的数量增加是随机的。

8. 服务器的安全性测试

服务器部件冗余显得非常重要，因为服务器冗余是消除系统错误、保证系统安全和维护系统稳定的有效方法，所以冗余是衡量服务器安全性的重要标准。

8.2　网络本身的测试

8.2.1　网络测试的类型

测试方法从宏观角度来说，可以分为三类，即一致性测试、性能测试和功能测试，如图 8-6 所示。

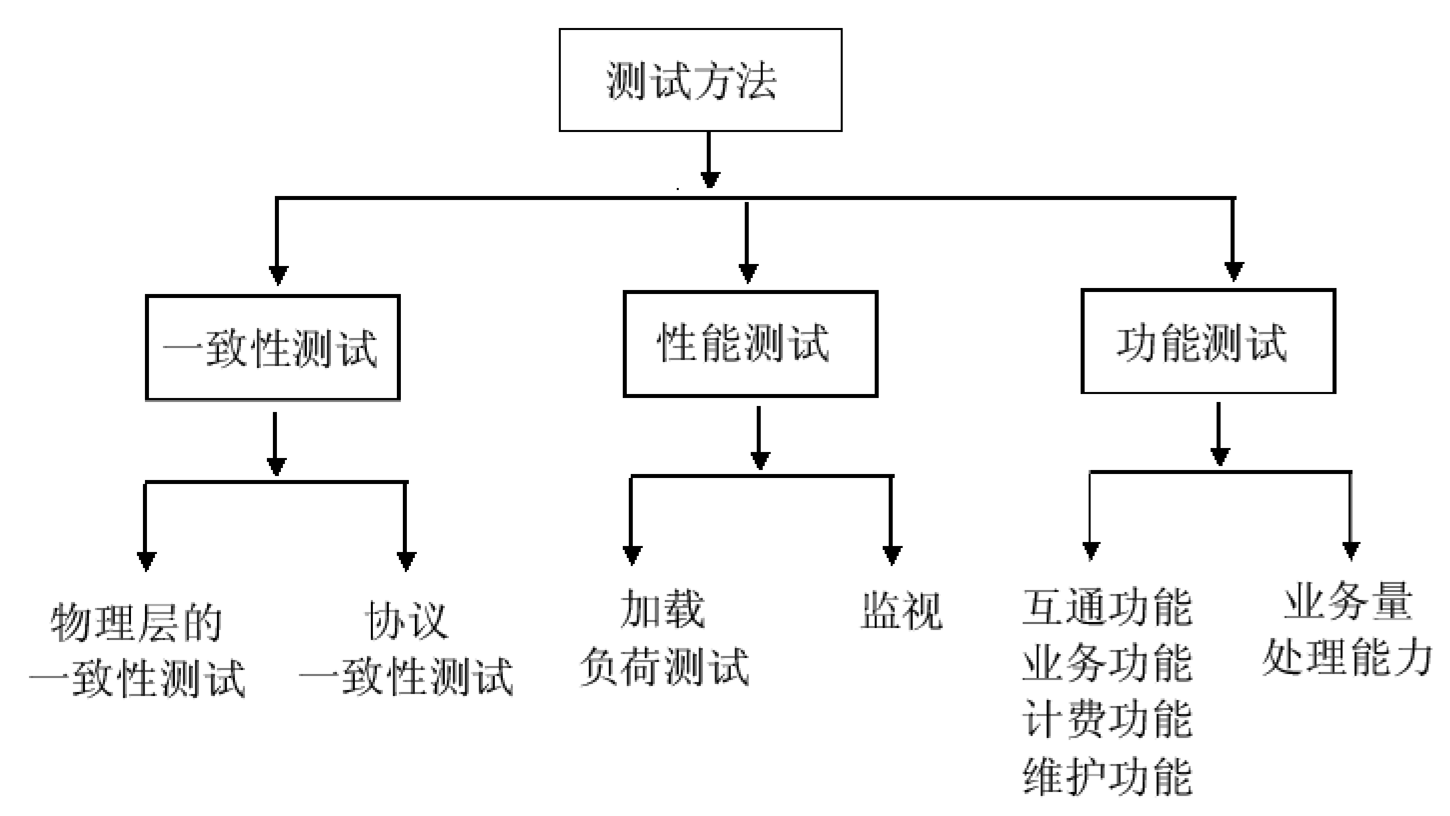


图 8-6　测试分类示意图

1. 一致性测试

通信协议的一致性测试是为了检验协议的实现与其协议标准是否一致而进行的测试。随着开放系统互联（OSI）标准的发展，国际标准化组织（ISO）也相应颁布了一系列协议的测试标准。目前，各国都相应地成立和正在筹建国家级的协议标准测试中心。

协议测试的目的，一方面是为了验证实现者所实现的协议软件是否符合标准规范的规定、是否具备协议的功能，另一方面也是为了当一个入网用户欲接入网络与网上其他用户通信时，保证各

通信用户之间能够按照统一的标准协调一致地完成通信功能，从而既不会由于协议执行差错引起通信失效，也不会使通信双方由于失序产生死锁或无法再同步。

（1）概念模型

一致性测试的概念模型如图 8-7 所示，其测试原理是测试器利用抽象服务原语（ASP）和协议服务单元（PDU）来观测和控制被测试对象（IUT）。测试器一般由对应于 IUT 上层接口的上测试器（UT）和对应于 IUT 下层接口的下测试器（LT）组成。测试时将测试器连接到 IUT 上，利用测试协调协议通过 UT 检测 IUT 的服务功能，通过 LT 检验协议。实际上，由 IUT 直接连接 UT 和 LT 的情况只是一种特例。一般是在 IUT 的上面装入上测试器，将 UT、IUT 和下面的设备合在一起构成被测试系统，再通过网络和一致性测试系统（CTS）连接起来进行测试。

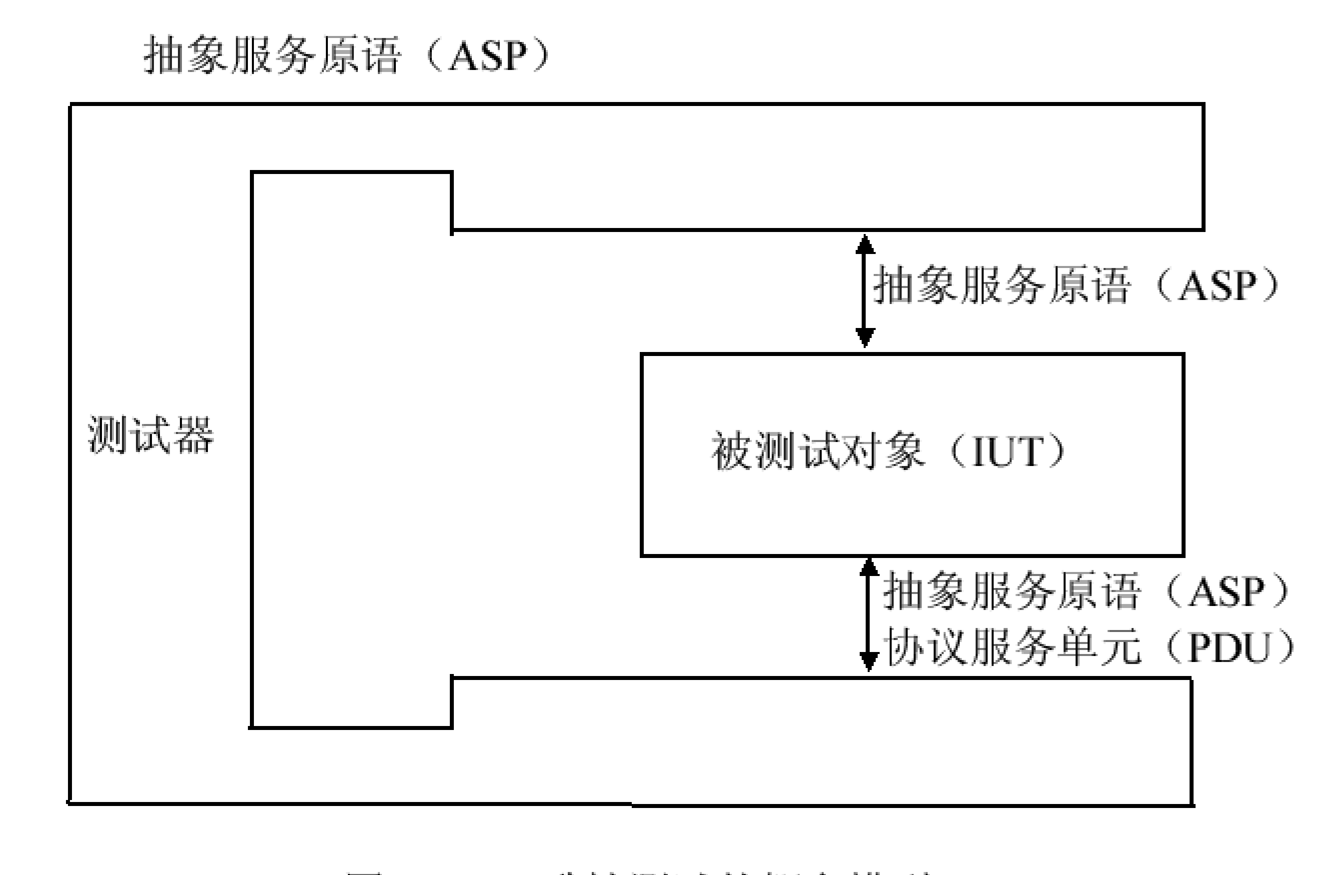


图 8-7　一致性测试的概念模型

（2）测试过程

一致性测试包括一致性测试组合的生成、一致性测试准备和一致性测试实施三个阶段。测试组合是由测试群、测试项目、测试步和测试事件构成。测试事件是最小的测试单位，测试步用于测试项目的分类和模块化。测试项目是实施的某一具体过程。测试群则是将相关联的测试项目进行分组，建立具有共同测试目的的测试项目集合。测试组合是根据被测对象层的协议规范和服务约定，提出静态一致性要求和动态一致性要求。静态一致性要求定义了协议实现所包含的最小能力，如协议的级别、可选项的定义等；动态一致性要求则定义了协议实现在通信所允许的行为，它们构成了协议的主体，是协议实现所具有的最大能力。一致性测试准备是进行一致性测试之前测试机构同 IUT 实施者间交换信息的过程。一致性测试实施主要完成静态一致性评价、测试选择和参数设定、基本互联测试、能力测试、动态测试、结果分析和最终一致性评价的功能，并生成系统一致性测试报告和协议一致性报告。

（3）测试方法

一致性测试的概念性测试结构，即所谓的“黑盒子”测试法。X.290 建议为 IUT 定义 4 种抽象测试方法，其中一种是内部的，其他三种是外部的，即下测试器远离被测系统，并通过链路或网络与之相连。这 4 种方法分别是本地测试法、分散测试法、协调测试法和远程测试法。使用时应根据

协议的层次、特性及测试器和 IUT 实现方面的制约等，分别选择相应的测试方法。一般对低层协议（网络层以下）多使用本地测试法、协调测试法和远程测试法。对高层协议多使用协调测试法、远程测试法和分散测试法。

对于 ISDN 用户-网络接口的协议测试，以 ISDN 终端为被测实体，为了能够使测试系统达到实用化，应该在三种外部测试法（即分散、协调和远程测试法）中进行选择。用测试系统模拟 ISDN 用户-网络接口网络侧的设备，通过 S/T 接口与被测终端进行数据交换。分散测试法和协调测试法均采用上测试器，即测试器需要直接介入各种终端的二、三层软件系统，在实践上难以实现，而远端测试法不包含实际的上测试器，对其进行改进后，使得在实际测试过程中，被测终端设备能够根据测试器的要求，配合测试系统完成例如摘机、挂机等高层的用户功能。这个要求对于实际被测系统来说是完全可以实现的，所以可以采用远程测试法进行 ISDN 协议的一致性测试。

（4）一致性测试的标准化

国际标准化组织进行了协议一致性测试的标准化研究，并已制订出 ISO9646（OSI 一致性测试方法论和框架）国际标准。1988 年 CCITT 制定出相应于 ISO9646 的 X.290 系列建议，两者的对应关系如表 8-1 所示。

表 8-1　ISO9646 和 CCITT X.290 系列建议的对应关系

一致性测试方法和框架	ISO9646	CCITT X.290 系列建议
一般概念	ISO9646-1	
抽象测试组合规范	ISO9646-2	CCITT X.291
TTCN	ISO9646-3	
测试实现	ISO9646-4	CCITT X.293
对测试机构和委托单位的要求条件	ISO9646-5	CCITT X.294

有关 ISDN 协议一致性测试的标准化工作在欧洲电信标准委员会（ETSI）的领导下取得了重要的战果。最为人熟知的标准是 ISDN 用户-网络接口终端侧的一致性测试规范 NET3 和 NET5。NET3 主要规范了 ISDN 基本接入（2B+D）接口的二层和三层的测试；NET5 规范了 ISDN 一次群接入（30B+D）接口的二层和三层的测试。国际电联也有相应的建议，已经颁布的 Q.921bis，为 ISDN 用户-网络接口的链路层制订出了一致性测试的标准。

2. 性能测试

性能测试通常检查被测实体的性能参数是否符合要求。性能测试一般不像一致性测试那样具有一套标准化的测试规范，但是需要完成对被测实体必要的一般性能的测试和服务质量的测试。目的是发现与规范要求不同的问题所在。性能测试的参数根据被测实体的不同而不同。以 ISDN 交换机为例，需要进行业务量加载的试验。虽然一般程控数字交换机均有自检功能，但仍需要进行充负荷的性能检测，以保证交换机在各种业务量情况下的正常操作。在定义加载测试时应该考虑：ISDN 用户线的业务量特征；在一定时间内可以规定所要求的试呼数量和在给定的被测网络资源中平均分配负荷。

这些加载测试的数据经过统计计算后，可以得到交换机的服务质量，以检查是否符合用户的要求，一般包括呼叫成功率、网络响应时间、呼叫连接建立时延、误码率和计费差错等。

3. 功能测试

功能测试一般检查被测实体是否能够完成技术规范所要求的功能，例如，ISDN 的互通功能，要求 ISDN 交换机完成 ISDN 与电话网的互通、与分组网的互通、与局域网的互通、与帧中继网的互通等。此外对 ISDN 交换机所要求的功能还有为用户提供各种补充业务的功能、各种信令的配合功能（包括 D 通路信令与 ISDN 局间信令的配合、D 通路信令与中国 1 号信令的配合、D 通路信令与 7 号信令（电话部分）的配合，以及 ISDN 局间信令与中国 1 号和 7 号信令（电话部分）的配合等）。ISDN 的交换机还应该完成故障的恢复功能和维护测试功能等。对这些功能的检测是非常必要的，协议分析仪是检测 ISDN 信令和协议的重要工具。

8.2.2 网络测试的内容

网络的基本测试内容主要包括对 ISDN 用户-网络接口的测试、用户线的测试、网上 ISDN 设备的测试、ISDN 网络的性能（业务及互通功能）的测试。从网络总体来讲，需要进行单机测试、终端与交换机的系统测试，最后进行 ISDN 的网络测试。

单机测试主要包括以下内容：

- 各类终端（含适配器）的物理层试验、用户信令二层和三层协议的试验、终端人-机接口的使用与评价、终端操作维护及其他辅助性能的试验与评价。
- 用户传输段的试验，即 S 接口网络侧的物理层试验、U 接口的物理层试验、V 接口的物理层试验、整个数字段的误码率试验。
- ISDN 交换机和用户交换机的试验，即交换机 V 接口物理层性能的测试、交换机连续误码性能的测试、交换机用户信令二层和三层协议的测试和操作维护、其他辅助性能的试验与评价。
- 接入分组数据网的接入单元（AU）的试验，即在 S 接口的物理层试验，包括对 X.25 信号结构正常插入 B 道路的验证、S 接口信令二层和三层的协议试验、R 接口对 X.25 各层协议的试验和操作维护、其他辅助性能的试验与评价。

终端与交换机的系统测试是将终端、网络终端和交换机连接在一起试验系统的工作及性能状况。

ISDN 的网络测试主要包括单局测试、局间测试和互通测试。

单局测试的主要内容是本局 ISDN 呼叫的测试，包括兼容终端间的呼叫测试、不兼容终端间的呼叫测试、终端选择的测试和 ISDN 用户与非 ISDN 用户间的呼叫互通测试。单局测试还包括本局 ISDN 补充业务的测试和网络性能的测试，例如，本局从 S 接口到 S 接口全程误码性能的测试等。

局间试验主要测试局间开通 7 号信令的工作情况，包括 MTP 和 ISUP 的测试。局间试验还包括网络性能的试验，例如局间从 S 接口到 S 接口全程误码性能的测试等。ISDN 与分组数据网的互通试验、ISDN 与电话网的互通试验、全网 ISDN 业务的试验及网络性能的试验等。

虽然 ISDN 的测试内容较多，但是最基本的测试是物理层的测试、链路层的测试、网络层的测试和分组终端接入 ISDN 的测试。

8.2.3 网络测试的方式

网络测试方式主要是指对 ISDN 设备进行测试的方法，就协议测试来讲，一般有以下几种方式。

1. 协议监视

协议监视测试是把协议分析仪跨接在 ISDN 用户-网络接口上，实现监视 ISDN 终端侧和网络侧之间的信息交换，并记录与显示各层协议的状态以及接收、发送的信息内容。

采用协议分析仪进行监测时，用户可以根据被测的协议内容设置显示窗口，进行协议传送的单向监视；也可以根据被测的协议内容设置显示窗口，进行协议传送的单向监视或双向监视。可以对每层协议同时监视，也可以仅对其中的某层协议进行监视。在对某一层协议进行监测时，还可以选择监视的具体内容，以监测 ISDN 的第三层协议为例，可以监测消息类型、信息单元类型、信息参数和呼叫状态等。

2. 协议的模拟和仿真

协议测试一般分为仿真和模拟两类。

（1）协议的仿真

仿真测试是将协议分析仪作为 ISDN 交换机或 ISDN 终端与被测实体相接，这样被测实体可以与协议分析仪直接进行信息交换。协议分析仪可以仿真呼叫的建立、保持和释放连接的全部过程。在这类协议仿真测试时，协议分析仪的仿真软件能够自动完成呼叫状态转移的功能。用户可以检测在每一个协议状态下应该发生的事件。协议的仿真测试主要是检测被测实体的完整的协议处理过程。

（2）协议的模拟

协议的模拟与仿真测试的区别在于模拟没有呼叫建立、保持及拆线的完整的状态转移机制。通常，由协议分析仪厂家提供一些测试程序供用户使用。用户也可以根据需要测试内容，预先编制好一些仿真测试软件来进行测试。模拟测试的优点是可以检测被测实体对异常情况及差错状态的处理能力，例如，使用协议分析仪向被测实体发送一些差错信息进行测试。与仿真测试相比，模拟具有较强的可操作性和灵活性。

总之，协议仿真测试的优点是被测实体无需接入实际的网络，就可以检验其运行的正确性，因此协议仿真在开发和安装过程中起着重要的作用。

8.2.4 网络应用系统的测试

网络应用系统的测试需要重点注意如下内容。

1. 整网性能

测试整网性能时的重点注意事项如下：

- 端到端的吞吐量。

- 端到端的延迟。
- 端到端的丢包率。

2. 流媒体性能

测试流媒体性能时的重点注意事项如下：

- 流尝试的 Session 总数。
- 流成功的 Session 数。
- 流失败的 Session 数。
- 平均流时间。
- 估计服务器响应时间。
- 抖动小于 50 ms 的流数。
- 抖动 51~100 ms 的流数。
- 抖动 101~300 ms 的流数。
- 最大网络流量 Rx（Mbps）。
- 最大网络流量 Tx（Kbps）。

3. 网管系统

测试网管系统时的重点注意事项如下：

- 设备管理。
- 流量管理。
- 日志管理。

4. 网络性能管理

测试网络性能管理时的重点注意事项如下：

- 安全策略。
- 平台策略。
- 管理维护中心策略。
- 系统服务器区策略。
- 历史图像系统服务器区策略。

5. 视频及编码格式

测试视频及编码格式时的重点注意事项如下：

- 模拟格式。
- 制式。
- 接口。
- 数字格式。
- 视频编解码。
- 音频编解码。
- 物理接口。

- 视频传输接口。
- 数字传输接口。
- 模拟接口。

6. 图像质量

测试图像质量时的重点注意事项如下：

- 传输网络。
- 同轴电缆。
- 光纤。
- 无线。
- 图像质量。
- 随机信噪比。
- 单频干扰。
- 电源干扰。
- 脉冲干扰。
- 清晰度测试。

7. 核心平台系统管理功能

测试核心平台系统管理功能时的重点注意事项如下：

- 监控功能。
- 录像及回放功能。
- 报警管理。
- 图像智能分析。
- 灾备恢复。
- 网络可用性。
- 网络流量与拥塞控制。
- 链路冗余。
- 设备冗余。

8. 中心平台

测试中心平台时的重点注意事项如下：

- 大屏幕显示系统。
- 中央控制系统。
- 综合指挥系统。
- 综合保障系统。
- 办公系统。

8.2.5 网络性能测试的环境

网络性能测试环境将根据实际网络环境来确定。

1. 性能测试指标

测试性能指标时的重点注意事项如下。

- 吞吐量（Throughput）：吞吐量是指在不丢包的情况下能够达到的最大包转发能力，一般以所能达到线速的百分比（或称通过速率）来表示。
- 延迟（Latency）：延迟是指测试仪发出数据包到经过网络后收到该数据包的时间间隔，单位为 10^{-6} 秒。
- 延迟抖动：传输中延迟分布的范围。
- 每秒新建连接数（Transactions Per Second）：每秒新建连接数是指一定流量下通过网络的主机之间每秒所能建立的最大连接数量。
- Http 响应时间（Http Response Time）：指用户从开始发出请求到得到服务器的完整的页面内容的平均响应时间，它能够很直接地反映客户端通过 Http 访问 Web 并调用后台数据库给出响应的时间，能有效反映出被测产品的性能（包含不同大小的页面文件），单位为 10^{-3} 秒。

2. 性能测试方法

性能测试分为两大类：网络层性能和应用层性能。网络层性能主要反映网络数据包转发能力、转发时间等，是应用层性能的基础。应用层性能是模拟用户的正常应用行为，测试其性能，能够直接反映出网络应用的实际性能。

在进行性能测试前，需要提前做好设备配置备份的准备工作。

8.2.6 网络应用系统的测试阶段划分

为保证测试执行的顺利进行，将网络应用系统的测试总体划分为 4 个阶段，即测试方案制定阶段、测试工具部署和准备阶段、测试实施阶段、报告分析阶段。网络应用系统测试实施各阶段的工作说明如表 8-2 所示。

表 8-2 网络应用系统测试实施各阶段的工作说明

阶段划分	测试
测试方案制定阶段	项目测试需求分析 提出测试需求 编制本项目实施的测试方案
测试工具部署和准备阶段	确定网络环境拓扑图 研究设备接入接口及地址分配 提供测试中涉及到的设备的配置信息或只读权限 必要时增加访问控制设备的安全策略

（续表）

阶段划分	测试
测试实施阶段	对不同区域间的吞吐量、延迟、每秒新建连接数、响应时间进行测试 对系统安全进行测试 对安全策略进行测试 详细记录测试结果 对数据结果进行初步整理分析，如出现数据不完整等情况可及时进行补测
报告分析阶段	测试问题报告 根据测试情况编写测试报告 提交报告审核 出具最终测试报告

对表 8-2 的说明如下。

- 测试方案：测试方案是指在正式测试实施开始前，对测试项目所作的一个测试计划和执行方案，主要包括测试目的、评测依据、评测管理、评测内容及方法、测试配合要求、测试结果、测试环境要求以及项目输出成果等。
- 测试问题报告：测试问题报告是指在测试实施完成后，测试工作组提交的一个所发现的问题的报告，主要内容包括问题的描述及分析等。
- 测试报告：测试报告是指由测试工作组提交的最终测试结果报告，主要内容为整个网络的综合评价、详细测试结果描述以及测试环境描述等。

8.2.7　网络应用系统的主要测试设备

网络应用系统的主要测试设备如表 8-3 所示。

表 8-3　网络应用系统测试的主要测试设备

序号	工具名称	类别	用途
1	SmartBits 6000B	数据网络测试仪	测试网络层性能，主要包括吞吐量、延迟、丢包率等指标
2	SmartBits 600B	数据网络测试仪	测试网络层性能，主要包括吞吐量、延迟、丢包率等指标
3	LAN 3321A	千兆测试板卡	测试性能
4	XFP-3731	万兆测试板卡	测试核心交换机性能
5	CDMA 时间同步模块	时间同步模块	用于在测试整网性能时的时间同步
6	Avalanche 2500C	网络应用层仿真及性能测试仪	测试应用层性能，主要包括每秒新建连接数和响应时间、最大并发连接数、可用带宽、流媒体性能等指标
7	Reflector 2500C	网络应用层仿真及性能测试仪	测试应用层性能，主要包括每秒新建连接数和响应时间、最大并发连接数、可用带宽、流媒体性能等指标
8	AURORA-200 漏洞扫描	漏洞扫描器	扫描网络内部各节点存在的漏洞



如果还需要其他测试设备，可根据测试情况进行调整和追加。

8.3 软件安装的测试

软件的安装测试可分为共享软件安装测试和用户应用系统软件安装测试。

8.3.1 共享软件安装测试

共享软件的安装可分为需要安装和不需要安装直接运行两种方式。共享软件测试流程如图 8-8 所示。

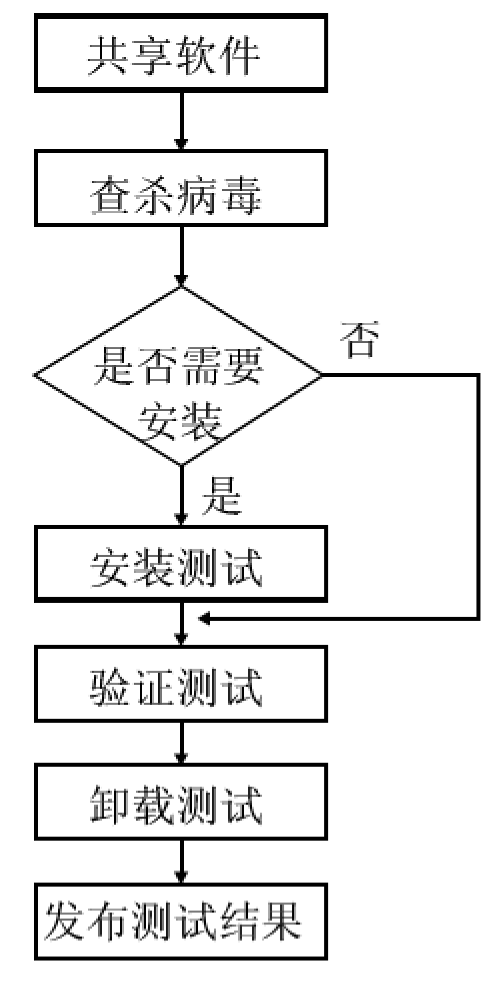


图 8-8 共享软件测试流程图

其中，对图 8-8 中出现的重点环节的说明如下。

1. 是否需要安装

若共享软件需要安装，则依据安装向导进行安装，验证被测软件是否对要安装的插件做出明确的提示。

若共享软件不需要安装，则共享软件跳过安装测试。

2. 验证测试

验证共享软件时需要重点注意如下内容：

- 是否弹出强制性广告。
- 共享软件运行时是否有浮动的广告。
- 共享软件是否强制用户单击了广告之后才能使用。

- 程序能否调用所有的已经安装的插件增加其自身的功能。
- 打开应用程序，验证所有已安装的插件，查找出不增强软件自身功能的插件，并做好记录。
- 是否开启了后门程序（间谍软件）收集用户信息。
- 查看系统进程中是否有可疑进程。
- 在系统服务中是否增加了莫名的服务。
- 安装防火墙记录软件访问网络的信息。
- 查看浏览器是否被劫持。
- 是否安装了新的插件。
- 在正常访问网站时是否被跳转到恶意网页。
- 查看 IE 浏览器主页/搜索页等是否被修改。
- 系统是否安装了流氓插件。

3. 卸载测试

测试共享软件能否成功地卸载时，需要重点注意如下内容。

- 自带卸载程序：查看共享软件是否具有自带的卸载软件，有则利用自带的卸载程序进行卸载，卸载完成后查看系统是否把共享软件及其附带的插件成功卸载。
- 利用 Windows 自带的添加/删除程序卸载。

4. 发布测试结果

发布共享软件的测试结果如表 8-4 所示。

表 8-4　发布共享软件的测试结果

软件名称			描述
评测情况	评测日期		填写测试共享软件的日期
	测试环境		填写软件的操作系统
	软件版本		填写软件的版本号
	安装文件类型及大小		填写软件的安装文件的类型及大小（精确到字节）
	有无病毒		
	有无可选插件		如果有则列出可选插件，并附说明
	有无强制安装插件		如果有则列出强制安装的插件
	有无强制广告		如果有则附说明
	有无后门程序		如果有则列出后门程序
	能否完全卸载		如果有插件无法完全卸载
	自带卸载程序		
	用 Windows 自带的添加/删除程序卸载		

8.3.2　用户应用系统软件安装测试

用户应用系统软件安装测试是指按照软件产品安装手册或相应的文档，进行一步一步地操作，从而完成安装的过程所进行的测试。安装测试不可忽略，安装测试是重要的一个测试阶段。

用户应用系统软件安装可以分为以下几种。

- 全新安装：安装的软件包是完整的，包含了所有的文件。
- 客户端软件安装。
- 服务器安装。
- 整个网络系统安装。
- 升级版本安装：部分文件构成的软件包。
- 补丁式安装：很小的改动或很少文件的更新，软件版本不变。

对用户应用系统软件进行安装测试时，需要重点注意的内容如下。

- 用户的使用环境不同、设置或配置不同，将影响应用系统软件安装的准确性。
- 安装的文档是否准确，文档安装时，必须一步一步地完全按照文档要求去做，不可忽略。
- 查看信息查询功能。
- 查看信息发布功能。
- 查看信息统计功能。
- 查看信息管理功能。

第9章　性能测试技术

对于软件应用系统，仅仅从功能上满足用户的需求是不够的，还需要从性能方面更好地满足客户的需要。

性能尤其对于实时系统、嵌入式系统和在线服务系统要求更高些。这就要求我们做好系统的性能测试，以保证系统的高性能、高可用性。性能测试，一般都通过测试工具来模拟人为的操作而进行。性能测试的重点在于测试环境的建立、前期数据的设计与后期数据的分析。

本章重点讨论以下内容：

- 性能测试概述。
- 性能测试的实例剖析。

9.1　性能测试概述

9.1.1　性能测试的分类

性能测试主要分为基本性能测试和高级性能测试两个方面。

1. 基本性能测试

基本性能测试的主要内容包括：安全可靠性测试、资源占用率测试、兼容性测试、易用性测试、用户文档测试、效率测试、可扩充性测试。

（1）安全可靠性测试

安全可靠性测试的示意如表9-1所示。

表9-1　安全可靠性测试

序号	测试项目	描述	测试结果
1	用户权限限制	考察对不同的用户权限的限制情况	符合/基本符合/不符合
2	用户和密码封闭性	对于相应用户及密码进行次数限制	符合/基本符合/不符合
3	屏蔽用户操作错误	考察对用户常见的误操作的提示和屏蔽情况	符合/基本符合/不符合
4	错误提示的准确性	对用户的错误提示的准确程度	符合/基本符合/不符合
5	错误是否导致系统异常退出	有无操作错误引起系统异常退出的情况	符合/基本符合/不符合

（续表）

序号	测试项目	描述	测试结果
6	数据备份与恢复手段	系统是否提供备份及恢复功能、备份手段如何，如是否对备份数据加密、压缩	符合/基本符合/不符合
7	输入数据有效性检查	系统对数据录入的有效性检查	符合/基本符合/不符合
8	留痕功能	系统是否有操作日志，操作日志记录的操作情况的全面性和准确性，是否包括主要要素，如操作员、操作日期、使用模块等	符合/基本符合/不符合
9	异常情况的影响	在程序运行过程中进行掉电试验，考察数据和系统的受影响程度，若受损，是否提供补救工具、补救的情况如何	符合/基本符合/不符合
10	数据传输安全性	对于有特殊安全要求的数据传输，应对传输的数据进行必要的加密处理，使用的算法应符合国家规定	符合/基本符合/不符合

（2）资源占用率测试

资源占用率测试的示意如表 9-2 所示。

表 9-2　资源占用率测试

序号	测试项目	描述	测试结果
1	软件安装所占用的硬盘空间	考察软件安装所占用硬盘空间	符合/基本符合/不符合
2	模块装载后内存占用量（包括虚存）	考察模块装载后内存占用量（包括虚存）	符合/基本符合/不符合
3	模块卸载后内存释放率（包括虚存）	考察模块卸载后内存释放率（包括虚存）	符合/基本符合/不符合

（3）兼容性测试

兼容性测试的示意如表 9-3 所示。

表 9-3　兼容性测试

序号	测试项目	描述	测试结果
1	软件兼容性	软件的适用平台	符合/基本符合/不符合
2	硬件兼容性	硬件平台的配置要求	符合/基本符合/不符合

（4）易用性测试

易用性测试的示意如表 9-4 所示。

表 9-4　易用性测试

序号	测试项目	描述	测试结果
1	易安装性	安装的难易程度，符合流行安装模式	符合/基本符合/不符合
2	用户界面的友好性	界面的简捷性如何	符合/基本符合/不符合
3	易学习性	相对一般操作人员来说，学习使用的难度如何、对操作人员有何要求	符合/基本符合/不符合

（续表）

序号	测试项目	描述	测试结果
4	易操作性	操作的难易程度	符合/基本符合/不符合
5	联机帮助丰富性	考察联机帮助的准确性、全面性，在关键操作时使用联机帮助的方便性	符合/基本符合/不符合

（5）用户文档测试

用户文档测试的示意如表 9-5 所示。

表 9-5　用户文档测试

序号	测试项目	描述	测试结果
1	用户手册的完整程序	用户手册内容的全面性、完整性	符合/基本符合/不符合
2	用户手册的描述与软件实际功能的一致性	手册与软件实际功能的一致程度	符合/基本符合/不符合
3	用户手册的易理解程度	用户手册对关键重要的操作有无图文说明，例图的易理解性如何	符合/基本符合/不符合
4	用户手册的印刷与包装质量	用户手册包装的商品化程度与印刷质量	符合/基本符合/不符合
5	用户手册提供学习操作的实例	对主要功能和关键操作提供的应用实例有多少、实例的详细程度如何	符合/基本符合/不符合

（6）效率测试

效率测试的示意如表 9-6 所示。

表 9-6　效率测试

序号	测试项目	描述	测试结果
1	通信效率	网络负载、吞吐率、利用率、响应时间、延迟等	符合/基本符合/不符合
2	设备效率	CPU 占用率、内存占用率、磁盘占用率、输入输出效率等，包括软件在不工作状态下对于硬件资源的占用情况和进行业务处理过程中对于硬件资源的占用情况	符合/基本符合/不符合
3	执行效率	典型业务操作的执行效率，例如关键的查询、统计等的响应时间	符合/基本符合/不符合

（7）可扩充性测试

可扩充性测试的示意如表 9-7 所示。

表 9-7　可扩充性测试

序号	测试项目	描述	测试结果
1	与异种数据接口	有无与其他数据的接口	符合/基本符合/不符合
2	是否能扩充功能模块	能否根据用户要求扩充功能模块	符合/基本符合/不符合

2. 高级性能测试

高级性能测试的主要内容包括：并发性能测试、系统资源监控测试、大数据量测试、速度测

试、疲劳测试等内容，其中，并发性能测试是重点。

（1）并发性能测试

并发性能测试的过程是一个负载测试和压力测试的过程，即逐渐增加负载，直到系统的瓶颈或者不能接收的性能点，通过综合分析交易执行指标和资源监控指标来确定系统并发性能的过程。

并发性能测试及系统资源监控使用自动化负载测试工具及监控工具。

测试案例：中间件应能满足一定数量的前台客户端同时办公的需要。

测试内容与监控指标：负载压力测试、模拟不同数量并发用户测试。

模拟不同数量并发用户执行关键业务，测试系统能够承受的最大并发用户数，主要监控指标如下：

- 每分钟事务处理数（Transaction Rate）：不同负载下每分钟成功完成的事务处理数。
- 响应时间（Response Time）：服务器对每个应用请求的处理时间，单位为秒，该项指标反映了系统事务处理的性能，具体包括以下几项参数：Min 表示最小的服务器响应时间；Mean 表示平均的服务器响应时间；Max 表示最大的服务器响应时间；StdDev 表示事务处理服务器响应的偏差，值越大，偏差越大；Median 表示中值响应时间；90％表示 90％事务处理的服务器响应时间；虚拟并发用户数（Total Virtual Users）表示测试工具模拟的用户并发数量。

（2）系统资源监控测试

在进行负载压力测试的同时，利用测试工具对数据库服务器、Web 服务器、应用服务器、认证及授权服务器上的操作系统、数据库以及中间件等资源进行监控。

监控系统资源指标，在测试中，根据测试需求以及测试环境的变化，选取有意义的数据进行分析。

（3）大数据量测试

测试案例：考虑系统未来发展需要的存储空间，添加大数据量测试。

测试内容：主要包括两个方面的内容，一是单独的数据量测试；二是与并发性能测试相结合的综合测试。测试数据的准备借助于测试数据管理与生成工具，例如 FileAid。

（4）速度测试

测试案例：磁盘访问速度、备份速度以及网络办公系统运行速度等。

测试内容：主要是人工测试。

（5）疲劳测试

通常是采用系统稳定运行情况下能够支持的最大并发用户数，持续执行一段时间业务，通过综合分析交易执行指标和资源监控指标来确定系统处理最大工作量强度性能的过程。

9.1.2　性能测试的目的

产品性能决定产品是否达到产品规格书所要求的性能指标，性能测试的目的如下：

- 系统是否满足预期的性能要求。
- 作为对系统进行调试的参考。
- 系统的可扩展性。
- 利用性能测试手段发现系统存在的问题。
- 提供部署方案的参考。

9.1.3　性能测试的指标

性能测试的指标一般通过两种形式描述：产品需求指标和系统性能指标。

1. 产品需求指标

产品需求指标是指以下内容：

- 给出产品性能的主要指标，如在 100000 记录中查询一个特定数据的时间为 0.5 秒。
- 以某个已发布的版本为基线，例如，比上一个版本的性能提高 30% ~ 50%。
- 和竞争对手的同类产品进行比较。

2. 系统性能指标

系统性能指标是指以下内容：

- CPU 利用率。
- 内存占用率。
- 磁盘 I/O。
- 响应时间。

9.1.4　性能测试的内容

性能测试方案应包含以下内容：

- 对软件系统架构的分析（了解输入、输出数据的类型、数据量）。
- 性能测试组网图（网络环境说明）。
- 硬件环境说明。
- 测试范围、目的与方法。
- 性能测试工具的选型。
- 测试的启动/退出条件。
- 测试执行及测试结果分析。

9.1.5　性能测试的策略

性能测试的策略一般从需求设计阶段开始讨论制定，策略的内容决定着性能测试工作投入多

少资源、什么时间开始实施等后继工作如何安排。决定性能测试的策略的主要因素如下。

1. 指标性能

系统在需求分析、设计阶段和产品说明书等文档中明确的提出性能指标，这些指标是性能测试要完成的工作。

2. 独立业务性能测试

独立业务主要是指软件产品的模块具有独立业务功能，在需求阶段就可以确定，要单独测试其性能。

3. 业务性能组合测试

应用类软件系统通常不会使所有的用户只使用一个或者几个核心业务模块，可能是对多个业务进行组合使用，对多个业务进行组合性能测试。由于组合业务测试是最能反映用户使用系统的情况，因而业务性能组合测试是测试的核心内容。

4. 疲劳强度性能测试

疲劳强度性能测试是在系统稳定运行的情况下模拟较大的用户数量，并长时间运行系统的测试，通过综合分析执行指标和资源监控来确定系统处理最大业务量时的性能，主要目的是为了测试系统的稳定性。

5. 大数据量性能测试

大数据量性能测试是为了测试系统的业务处理能力进行的。

大数据量性能测试可分为两种，第一种是针对某些系统存储、传输、统计查询等业务进行的大数据量测试，主要是测试数据增多时的性能情况；第二种是极限状态下的数据测试，主要是指系统数据量达到一定程度时，通过性能测试来评估系统的响应情况，测试的对象也是某些核心业务或者日常常用的组合业务。

6. 网络性能测试

网络性能测试主要是为了准确展示带宽、延迟、吞吐量、负载、瓶颈和端口的变化是如何影响用户的响应时间的。重点测试吞吐量指标，因为 80%的系统性能瓶颈是由吞吐量造成的。

9.1.6 性能测试的方法

性能测试的方法主要有：能力验证、规划性能、性能调优、压力加载、性能下降曲线分析等。

1. 能力验证

能力验证强调：系统具备的硬件设备、软件环境、网络条件、基础数据。能力验证将使用到可靠性测试、压力测试、失效恢复测试。

2. 规划性能

规划性能关心的是要求系统具有的性能，强调系统配置，使系统能够满足增长的用户数的需要等问题。规划性能将使用到负载测试、配置测试、压力测试。

3. 性能调优

性能调优关心的是要求系统确定基准环境、基准负载和基准性能指标；调整系统运行环境和实现方法；记录测试结果、进行测试分析。

4. 压力加载

压力加载主要强调以下内容：

- 稳定压力加载。一次性将负载加到某个水平，并持续一段时间。
- 逐渐加载或交替加载到某个负载水平。
- 峰谷测试。确定从系统高峰时间的负载转为几乎空闲、再攀升到高负载这样峰值交替情况下的系统性能状态/指标。

5. 性能下降曲线分析

性能下降曲线分析关心的是性能随着用户数的增加而出现下降趋势的曲线分析、查看性能下降的环境点与上下文，并确定性能阀值。性能曲线通过单用户区域、性能平坦区域、压力区域、性能拐点进行监控和分析。

9.2 性能测试的实例剖析

某部电子政务基础平台系统中 5 个重点业务应用系统的运行模式采用 B/S 结构，针对这种实际情况，现为某部电子政务基础平台的具体性能测试如下内容。

- 并发性能测试。
- Web 站点质量分析。
- 应用故障定位。
- 测试策略。

下面详细论述每一种测试策略。

9.2.1 并发性能测试剖析

测试的基本策略是自动负载测试，通过模拟成百或上千的用户执行关键业务，对应用程序进行测试。通过可重复的、真实的测试能够彻底地度量应用的可扩展性和性能，确定问题所在，并优化系统性能。

并发性能测试的目的主要体现在三个方面：

- 评价系统当前性能。
- 预测系统未来性能。

- 通过重复测试寻找瓶颈问题。

1. 评价系统当前性能

以真实的业务为依据，选择有代表性的、关键的业务操作设计测试案例，并且能够提供图文并茂的分析报告及结果，主要的测试指标如下。

- 每分钟交易数（Trans Rate）：负载测试过程中每分钟完成的交易数（成功），这项指标用来确定系统的实际交易负载。
- 交易响应时间（Response Time）：服务器对每个应用请求的处理时间，单位为秒，该项指标反映了系统事务处理的性能。

2. 预测系统未来性能

当要扩展应用程序的功能或者新的应用程序将要被部署时，负载测试会帮助确定系统是否还能够处理期望的用户负载。QALoad 并不需要调用到最终用户或其设备，它能够仿真数以千计的用户进行商业交易，并确定应用响应时间，使之符合产品的服务等级要求。

3. 寻找瓶颈问题

QALoad 录制/回放功能提供了一种可重复式的方法来验证负载下的应用性能，可以很容易地模拟数千个用户，并执行和运行测试。利用 QALoad 反复测试可以帮助充分地测试容量的问题，快速确认性能瓶颈并优化和调整应用。

并发测试使用的测试工具是 QALoad，它是由美国 Compuware（康博）公司开发的自动化负载测试工具。软件针对各种测试目标提供了 DB2、DCOM、ODBC、Oracle、NETLoad、Corba、QARun、SAP、SQL Server、Sybase、Telnet、TUXEDO、UNIFACE、WinSock、WWW 等不同的测试接口（Session），支持 Windows 和 Unix 测试环境。该工具还能在性能测试的全过程中，自动地完整记录服务器各项资源的消耗情况，为最终的性能分析和调整提供全面的、真实的数据。

针对某部电子政务基础平台的应用系统，具有很多需要保证能够承载并发压力的关键点，例如：

- 安全系统平台中的证书审核注册，测试是否满足总局最大数量用户正常访问的需求。
- 安全系统平台中的证书查询验证服务系统，测试系统是否满足基本的 LDAP 查询和 OCSP 查询并发数。
- 安全系统平台中的密钥管理系统，针对按实际需要配置的密钥管理系统，测试受理点连接数、签发在用证书数目、密钥发放并发请求数是否满足基本业务需求。
- 安全系统平台中的集中式授权服务，测试系统是否满足基本授权服务并发数。
- 应用系统中大量的查询操作也是主要的并发压力承受点。

9.2.2 Web 站点质量分析剖析

某部电子政务系统中 5 个重点应用系统的运行模式采用 B/S 结构，所以保证 Web 站点质量是至关重要的。利用站点测试工具，例如 Compuware 的 WebCheck 确保站点质量。测试过程中自动扫描 Web 站点不止 50 个的潜在问题类型，并且提供 19 个 HTML 报告。

Web 站点质量分析的重点如下：

- 检查内部和外部连接中成功和失败的连接点，包括网站上不用的文件、网页丢失的图像标题标签和属性标签等。
- 分析网站的结构，包括显示和某个 URL 相关的连接及按照标题、描述、作者、大小、最后修改时间、类型为 URL 连接分类等。
- 可以提供 HTML 报告，内容包括警告错误、失效的 URL、失效的页面等信息。

Web 应用功能测试也是非常重要的，可使用 Compuware 的 TestPartner 测试基于浏览器的 Web 应用（Internet Explorer and Netscape）、测试 COM 组件。

9.2.3 应用故障定位剖析

基于 Web 的分布式应用性能分析对保证系统正常运行非常关键。

利用 Compuware 的 Application Vantage，在真实网络环境下，基于 Web 和分布式的客户机/服务器应用，可以实现如下测试：

- 快速诊断、排除客户机/服务器、Web 和多层应用问题。
- 测试跨越多个网段的活动过程。
- 提供有关应用效率的重要统计数据。
- 建立多种报告，为不同的部门提供共同的观察问题视点，便于在应用开发者、网络管理员和数据库管理员之间的交流、沟通和协调。

9.2.4 测试策略剖析

为基础平台和应用系统提供的测试策略还包括：疲劳性能测试、速度测试、数据量测试、对主机和网络的监控测试、安全控制、文件系统测试、用户测试、内存测试、交换区 SWAP 测试、打印机测试、CPU 测试、进程测试、活动进程测试、对数据库及应用系统的监控等。

1. 疲劳性能测试

测试电子政务系统在一定时间段内持续执行某一业务的性能，主要考察交易执行的响应时间以及成功执行的交易数。

2. 速度测试

测试电子政务系统中某些业务点的执行速率，目的是保证满足需求以及提高系统的易用性。

3. 数据量测试

数据量测试主要是针对电子政务中某些业务，测试在大数据量压力下系统的性能以及数据库的性能表现。

4. 对主机和网络的监控

对主机和网络的监控主要包括以下方面的内容：

- 采集物理读/写和逻辑读/写的信息。
- 收集操作系统和其他平台上的磁盘忙信息。
- 监控 I/O。
- 确定网络是否过载。
- 显示 Internet 地址、输入和输出的包量、网络连接等信息。
- 显示错误包的比例。

5. 安全控制

进行安全控制时，主要查看以下方面的内容：

- 查找未设密码的用户。
- 检查所有设置成全局可写的文件。
- 列出所有访问根用户的信息。

6. 文件系统测试

进行文件系统测试时，主要查看以下方面的内容：

- 显示每个文件系统的使用率，检测文件系统空闲空间的大小。
- 剪裁文件系统，即删除指定的 Core 文件和其他文件。
- 显示文件系统的 mount on device、type、size 等内容。
- 可以监控特殊的文件系统，如 NFS、CD-ROM 等。
- 检测特定文件的存在及超出特定期限的文件存在。

7. 用户测试

进行用户测试时，主要查看以下方面的内容：

- 显示活动用户的 Sessions。
- 监控现在签到的用户情况。
- 显示每个用户的打印任务、用户的进程和用户的磁盘空间。
- 显示用户的用户数目和 Sessions 数目等内容。

8. 内存测试

进行内存测试时，主要查看以下方面的内容：

- 显示可用的内存数量。
- 决定当前的内存短缺量。
- 帮助分析内存问题。
- 显示内存的实存、所有虚存和 Kernel 的状态等信息。

9. 交换区 SWAP 测试

进行交换区 SWAP 测试时，主要查看以下方面的内容：

- 列出 SWAP 设备。
- 显示从二级存储区交换到主存储区的进程数量。

- 监控已用的和空余的交换区的比例。

10. 打印机测试

进行打印机测试时，主要查看以下方面的内容：

- 记录打印任务数。
- 显示打印机的 device、printer state、queue state、number of jobs in queue 等内容。

11. CPU 测试

进行 CPU 测试时，主要查看以下方面的内容：

- 记录 CPU 的使用率。
- 监测 CPU 参数。
- 显示 CPU 的总数。
- 显示 CPU 处理系统任务和完成用户任务的时间比例。

12. 进程测试

进行进程测试时，主要查看以下方面的内容：

- 识别消耗 CPU 最高的进程。
- 监控重要进程的状态。
- 标识内存使用最多的进程。
- 显示进程的进程数等信息。
- 识别僵死进程。

13. 活动进程测试

进行活动进程测试时，主要查看以下方面的内容：

- 通过允许指定需要监控的进程和监控时间（如某参数到达警戒值）来改进监控条件。
- 显示 kernel 的 kernel 名和创建日期等信息，监控 internal kernel tables 的使用率。

14. 对数据库及应用系统的监控测试

对数据库及应用系统的监控测试主要查看以下方面：

- 自动地管理数据库及数据库应用，监控数据库系统中关键的资源，一旦资源使用超过设定值则发出警报。帮助将关系数据库的运行调谐在最佳可靠性和最佳的性能表现上。
- 对数据库全天候不间断地监测。在监测到潜在的问题或者出错时发出警报和警告。
- 监测读写页面的使用情况，以便把对页面操作调谐到一个最佳水平。通过这一功能可以节省共享内存资源并且提高系统运行性能。
- 监控超出共享内存缓冲区的操作数，按照这个基准，可以对缓冲区进行额外的调整，以便更好地支持实际系统的运行需要，提高效率。
- 利用扩展的会话/用户检查以及参数控制功能发出警报，帮助发现过多的不合理顺序扫描操作。根据这些信息，可以分配附件的资源或者要求修改其应用程序以降低对系统资源的要求。可以制定更精确的资源容量计划，以便实现对现在和将来的业务运行需要。

- 跟踪逻辑日志的可用性，当达到预定的阀值时发出警告，以帮助尽快维护这一关键资源保持足够的可用空间。通过这项功能，可以保证数据库系统不因为日志空间问题而进入服务停顿状态。
- 监测上一轮询期间作业等待缓冲区的时间，帮助发现可能存在的并发性问题。通过一段时间的监测，可以得出系统的并发访问的需求和可用性基准参数。
- 跟踪共享内存中物理日志和逻辑日志的缓冲区的使用率，使我们能够准确地将其调谐到满足业务要求的最小状态，从而节省内存资源。有利于处理好内存容量计划、满足业务运行与增长的需要。
- 监控磁盘数据块的使用情况以及被频繁读写的热点区域。通过这些信息就可以较容易地平衡在磁盘上数据量的存储分配以及磁盘的 I/O 分配，有效地帮助我们作好磁盘容量规划并在当前的磁盘设备上有效地提高数据的读写效率。
- 根据用户事务或者表空间监控事务日志，通过这项监控，可以清晰地了解到重要作业对数据库存储、CPU 操作和内存使用的情况，精确测算和分析现有资源，使之能够更好地适应对数据库的持续要求。
- 监控数据库锁资源，监测关键业务的数据表的表空间增长，当空间低于预定的阀值时发出警告。通过这项功能，可以有效地安排关键业务数据的存储，并且有效地防止因为数据空间不足导致关键业务停顿。
- 监控碎片大小和扩展块大小。
- 监控 SQL 执行情况。
- 根据监控的数据库的参数配置、监控的数据和分析的结果调整数据库的参数，使数据库运行在最佳的服务水平上。

第10章　集成测试技术

集成测试也称组装测试，是在所有模块都通过了单元测试和子系统的功能测试成功的基础上，按照系统设计说明书的要求组合起来进行的测试。集成测试主要由系统设计人员、软件评测人员、开发人员共同完成。

本章重点讨论以下内容：

- 集成测试概述。
- 集成测试阶段工作。

10.1　集成测试概述

集成测试也称组装测试，属于白盒测试的范围。通过单元测试、功能测试的模块和子系统各部分工作是否达到或实现了相应技术指标及要求的功能是集成测试的工作。集成测试就是测试各个组件之间的配合情况，为系统测试提供一些基本保证。

在集成测试中可以弥补单元测试中没有测试到的Bug，也可以检查出单元测试中没有办法测试的功能，如前后台的集成之后的关联功能，对于这些有关联性功能的测试，单元测试是无能为力的，必须依靠集成测试来保证功能的完整性和正确性。与系统测试相比，集成测试从程序结构出发，目的性、针对性更强，发现问题的效率更高，较容易测试特殊的处理流程中存在的Bug。

集成测试的检测重点包括：子系统功能的关联性测试、链接完整性测试、数据和数据库完整性测试、功能测试、页面完整性测试等。

在实际操作中，通过单元测试和子系统功能测试的模块，在集成之后，仍可能会出现下列问题：

- 子系统接口的数据丢失。
- 单个模块的误差可以接受，但集成后会出现误差累积，从而影响其他模块，系统不能工作。
- 一个子系统功能对其他子系统造成有害的影响。
- 各个子系统集成起来没有达到预期的功能。

10.1.1　集成测试过程

集成测试过程如图10-1所示。

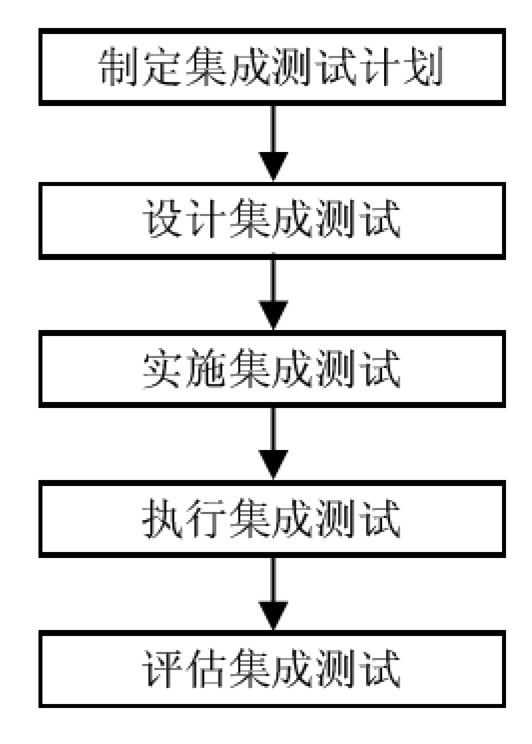


图 10-1　集成测试过程

集成测试相对来说比较复杂，而且对于不同的技术、平台和应用差异也比较大，更多是和开发环境融合在一起。集成测试所确定的测试内容，主要来源于设计模型。集成测试人员的工作过程如表 10-1 所示。

表 10-1　集成测试人员的工作过程

过程	工作内容	工作结果	担当人员和职责
制定集成测试计划	设计模型、集成构建计划	集成测试计划	测试设计员负责制定集成测试计划
设计集成测试	集成测试计划、设计模型	集成测试用例、测试过程	测试设计员负责设计集成测试用例和测试过程
实施集成测试	集成测试用例、测试过程、工作版本	测试脚本（可选）、测试过程（更新）	测试设计员负责编制测试脚本（可选），更新测试过程
		驱动程序或稳定桩	设计员负责设计驱动程序和桩，实施员负责实施驱动程序和桩
执行集成测试	测试脚本（可选）、工作版本	测试结果	测试员负责执行测试并记录测试结果
评估集成测试	集成测试计划、测试结果	测试评估摘要	测试设计员负责会同集成员、编码员、设计员等有关人员（具体化）评估此次测试，并生成测试评估摘要

10.1.2　集成测试方法

集成测试不是所有的代码编译通过就算是可以集成了，而是所有的模块、子系统能够正常运转。测试一般采用的方法是数据驱动或者桩驱动，因为集成测试不仅查看产品的表象，还要查看它的数据流，对数据流进行分析，从而掌握系统有什么不妥当的地方。

1. 数据驱动方法

数据驱动方法的实施方案有很多种，这里仅介绍自底向上集成测试、自顶向下集成测试、核心系统先行集成测试、高频集成测试。

在介绍数据驱动方法之前，先给出系统的各子系统功能联系图，如图 10-2 所示。

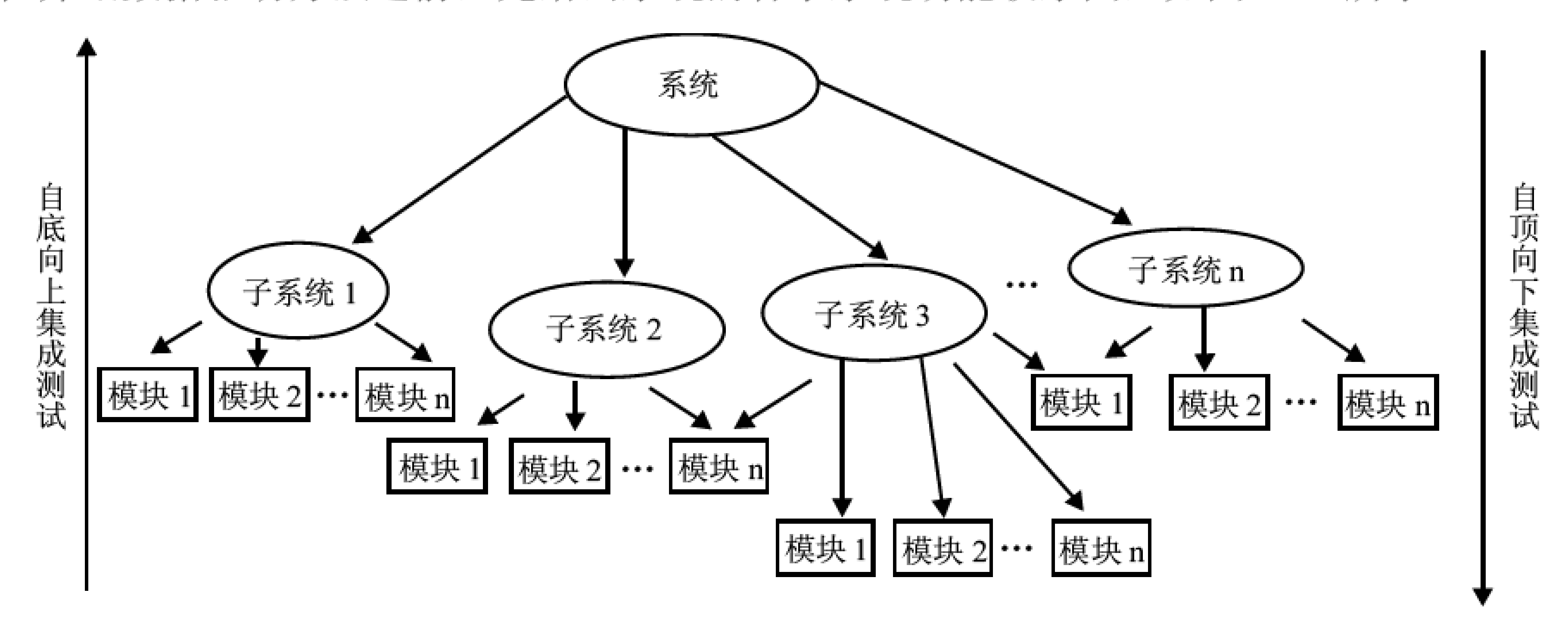


图 10-2　系统的各子系统功能联系图

下面将按照图 10-2 介绍数据驱动方法。

（1）自底向上集成测试

自底向上的集成（Bottom-Up Integration）方式是常用的方法，相关技术也较为成熟。其他集成方法都或多或少地继承、吸收了这种集成方式的思想。自底向上集成方式从系统结构中最底层的模块开始组装和测试。因为模块是自底向上进行组装的，对于一个给定的子系统，它的所有下属模块事前都已经通过了单元测试，不需要桩模块。自底向上集成测试的基本步骤如下。

01 按照概要设计规格说明，明确被测模块：在熟悉被测模块性质的基础上对被测模块进行分层，组织成实现某个子系统（功能）的模块群（Cluster），在同一层次上的测试可以并行进行，然后排出测试活动的先后关系，制定测试进度计划。

02 在 01 的基础上，按时间线序关系将软件单元模块集成为各个子系统，开发一个测试驱动模块，控制测试数据的输入和测试结果的输出。

03 检测各子系统是否能正常工作。需要成批输入数据，数据可分为如下几种：

- 没有错误的数据，以检查功能实现。
- 各种报错数据，以检查程序对错误数据的处理能力。
- 提供程序与程序交接口的数据（正确的、不正确的），以检查交接口的情况。

04 联机处理各子系统间的测试，其中联机处理各子系统间的测试有以下几类：

- 单任务单终端测试：一个终端上运行测试。
- 单任务多终端测试：同时在多个终端上运行测试。
- 多任务单终端测试：多任务在一个终端上运行测试。
- 多任务多终端测试：多任务在多个终端上运行测试。

05 联合测试：在 03、04 测试的基础上，还需要进行成批处理与联机处理的联合测试。联合测试成功则自底向上集成测试工作宣告完成。

（2）自顶向下集成测试

自顶向下集成测试是构造程序结构的一种增量方式，它从主控模块开始，按照软件的控制层

次结构，以深度优先或广度优先的策略，逐步把各个模块集成在一起。深度优先策略首先是把主控制路径上的模块集成在一起，至于选择哪一条路径作为主控制路径，这多少带有随意性，一般根据问题的特性确定。自顶向下集成的优点在于：能尽早地对程序的主要控制和决策机制进行检验，因此较早地发现错误。缺点是：在测试较高层模块时，低层处理采用桩模块替代，不能反映真实情况，重要数据不能及时回送到上层模块，因此测试并不充分。

自顶向下集成测试的基本步骤如下。

01 以主控模块作为测试驱动模块，把对主控模块进行单元测试时引入的所有桩模块用实际模块代替。

02 依据所选的集成策略（深度优先或广度优先），每次只替代一个桩模块。

03 每集成一个模块立即测试一遍。

04 只有每组测试完成后，才着手替换下一个桩模块。

05 为避免引入新错误，必须不断地进行回归测试（即全部或部分地重复已做过的测试）。

06 从 01 ~ 05 循环执行上述步骤，直至整个程序结构构造完毕。

（3）核心系统先行集成测试

核心系统先行集成测试法的思想是：先对核心软件部件进行集成测试，在测试通过的基础上再按各外围软件部件的重要程度逐个集成到核心系统中，直至最后形成稳定的软件产品。核心系统先行集成测试方法对于快速开发软件是有效的，适合较复杂系统的集成测试，能保证一些重要的功能和服务的实现。缺点是采用此法的系统一般要明确区分核心软件部件和外围软件部件，核心软件部件、外围软件部件要有很高的耦合度。核心系统先行集成测试法对应的集成过程是一个逐渐趋于闭合的，代表产品逐步定型的过程。核心系统先行集成测试的基本步骤如下。

01 对核心系统中的每个模块进行单独测试，必要时可使用桩模块。

02 对于核心系统中的所有模块一次性集合到被测系统中，解决集成中出现的各类问题。在核心系统规模相对较大的情况下，也可以按照自底向上的步骤，集成核心系统的各组成模块。

03 按照各外围软件部件的重要程度以及模块间的相互制约关系，拟定外围软件部件集成到核心系统中的顺序方案，进行外围软件部件的集成。

04 在外围软件部件添加到核心系统以前，外围软件部件应先完成内部的模块级集成测试。

（4）高频集成测试

高频集成测试是指同步于软件开发过程，每隔一段时间对开发团队的现有代码进行的一次集成测试。高频集成测试在开发过程中能及时发现代码错误、直观地看到开发团队的有效工程进度，这对有效防止错误、及时纠正错误都有很大帮助。高频集成测试的特点是：集成测试次数频繁，人工的方法是不胜任的。集成测试的基本步骤如下。

01 选择集成测试自动化工具。

02 设置版本控制工具，以确保集成测试自动化工具所获得的版本是最新版本。

03 测试人员和开发人员负责编写对应程序代码的测试脚本。

04 设置自动化集成测试工具，每隔一段时间对配置管理库的新添加的代码进行自动化的集成测试，并将测试报告汇报给开发人员和测试人员。

05 测试人员监督代码开发人员及时关闭不合格项。

06 按照 03~05 步骤不断循环，直至形成最终软件产品。

2. 桩驱动方法

桩驱动需要编制桩模块（集成测试前要为被测模块编制一些模拟其下级模块功能的“替身”模块，以代替被测模块的接口，接收或传递被测模块的数据，这些专供测试用的“假”模块称为被测模块的桩模块）。

桩驱动方法的执行示意图如图 10-3 所示。

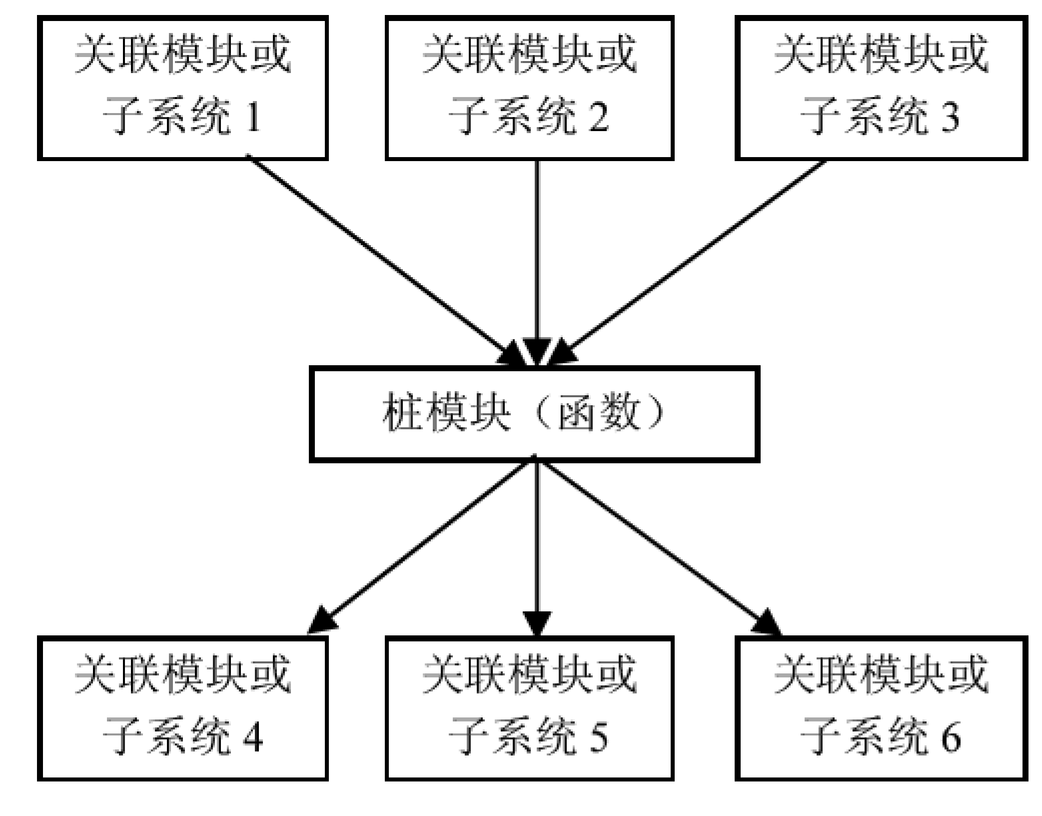


图 10-3 桩驱动方法

从图 10-2 中可以看出，如果中间的桩模块（函数）在单元测试时已经作过详细的测试，没有错误，那么关联模块或子系统 1、关联模块或子系统 2、关联模块或子系统 3 经过桩模块（函数）去关联模块或子系统 4、关联模块或子系统 5、关联模块或子系统 6 有错误，可能是输入的过程或者输出参数处理出错了。

桩驱动方法可以定义很多个桩，从而提高测试效率。

10.2 集成测试阶段工作

根据 IEEE 标准，集成测试可以划分为以下 4 个阶段：

- 计划阶段。
- 设计阶段。
- 实现阶段。
- 执行阶段（实施阶段）。

1. 计划阶段

在系统设计人员完成概要设计后，与软件评测部的人员一起进行集成测试计划阶段的工作。工作的内容如下：

- 按照系统需求规格说明书、概要设计说明书、系统开发计划进行集成测试计划阶段的工作。
- 确定出被测试对象和测试范围。

- 评估集成测试中被测试对象的数量及难度（即工作量）。
- 确定角色分工和任务。
- 标识出测试各阶段的时间、任务、约束等条件。
- 考虑风险分析及应急计划。
- 考虑和准备集成测试需要的测试工具、测试仪器、环境等资源。
- 考虑外部技术支援的力度和深度，以及相关培训安排。
- 定义测试标准。
- 写出集成测试计划。

2. 设计阶段

在系统设计人员完成系统详细设计后，与软件评测部的人员一起进行集成测试设计阶段的工作。工作内容如下：

- 按照需求规格说明书、概要设计说明书、集成测试计划、系统详细设计说明书进行集成测试设计阶段的工作。
- 被测对象结构分析。
- 集成测试模块分析。
- 集成测试接口分析。
- 集成测试策略分析。
- 集成测试工具分析。
- 集成测试环境分析。
- 集成测试工作量估计和安排。
- 写出集成测试设计（方案）。

3. 实现阶段

在编码阶段开始后，系统设计人员和软件评测部的人员一起进行集成测试实现阶段的工作。工作内容如下：

- 按照需求规格说明书、概要设计、集成测试计划、集成测试设计、入口条件、详细设计说明书进行集成测试实施工作。
- 按照集成测试用例设计、集成测试代码设计、集成测试脚本、集成测试工具要求写出集成测试用例、集成测试规程、集成测试代码、集成测试脚本。

4. 执行阶段

在单元测试已经完成后，系统设计人员和软件评测部人员一起进行集成测试执行阶段的工作。工作内容如下：

- 按照需求规格说明书、概要设计、集成测试计划、集成测试用例、集成测试规程、集成测试代码、集成测试脚本、集成测试工具、详细设计代码、单元测试报告执行测试。
- 执行集成测试用例。
- 回归集成测试用例。
- 撰写集成测试报告。

集成测试的报告如表 10-2 所示。

表 10-2 集成测试报告

项目名称		项目编号		
测试人		测试时间		
测试工具		测试环境		
操作性测试	问题类型：	□程序代码	□数据库	□项目文档
功能测试	问题类型：	□程序代码	□数据库	□项目文档
性能测试	问题类型：	□程序代码	□数据库	□项目文档
回归测试	问题类型：	□程序代码	□数据库	□项目文档
测试用例可加附页				
问题及影响描述、处理结果（可加附页）				
测试结论 测试负责人： 年 月 日		审核（项目经理）： 年 月 日		

第11章　系统测试技术

在完成集成测试阶段的工作后，建立的新系统已初步奠定了基础，新系统需要进行系统测试。系统测试是将已经通过集成测试的软件、计算机硬件、外设和网络等其他因素结合在一起，通过与系统的系统方案说明书、需求说明书相比较，发现系统与用户需求不符或矛盾的地方。新投入运行的系统是否正确无误，这一点极其重要，所以在系统实施运行前需要进行系统测试。未经周密测试的系统贸然投入运行，将会造成难以想象的后果。

本章重点讨论以下内容：

- 系统测试的主要内容和测试类型。
- 系统测试的过程。
- 系统测试的结果分析。
- 系统测试的文档资料。

11.1　系统测试的主要内容和测试类型

1. 系统测试的主要内容

系统测试的主要内容如下：

- 典型应用的反应时间。
- 客户端、服务器的 CPU、Memory 使用情况。
- 服务器的响应速度。
- 系统支持的最优负载数量。
- 网络指标。
- 系统可靠性测试。

2. 系统测试的测试类型

系统测试的测试类型一般包括如下内容：

- 功能测试。
- 性能测试。
- 负载测试。
- 强度测试。
- 容量测试。
- 安全性测试。
- 用户界面测试。

- 有效性测试。
- 配置测试。
- 故障恢复测试。
- 安装测试。
- 回归测试。

其中，功能测试、性能测试、负载测试、安全性测试、配置测试、有效性测试、故障恢复测试等在一般情况下是必须的，而其他的测试类型则需要根据软件项目的具体要求进行。

11.2　系统测试的过程

系统测试的一般过程如图 11-1 所示。

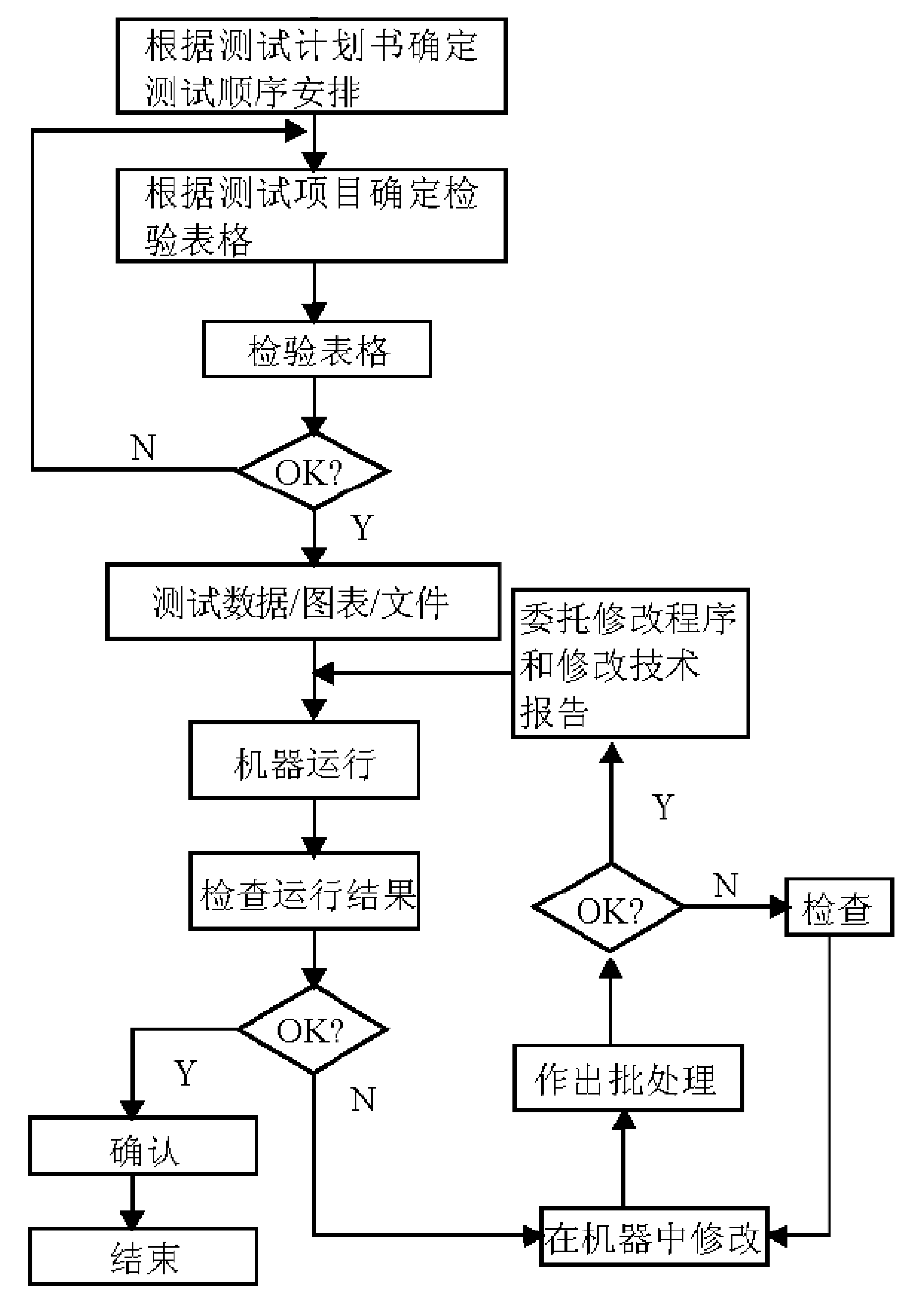


图 11-1　系统测试的流程

系统测试实际上是顺序实现 4~5 个步骤的序列。最初，测试集中在每个单独的模块，保证它作为一个单元测试；其次，必须将模块加以集中或装配，形成一个功能，系统有 n 个功能，功能和功能组装形成一个完整的功能；集成测试有检验和组装这两重含义，用于检验集成的所有元素配合

是否合理以及整个系统的性能和功能是否达到；有效性测试用于保证软件符合所有功能上和性能上的要求；最后进行系统测试。系统分为大系统、小系统，小系统测试实现的序列如图 11-2 所示，大系统测试实现的序列如图 11-3 所示。

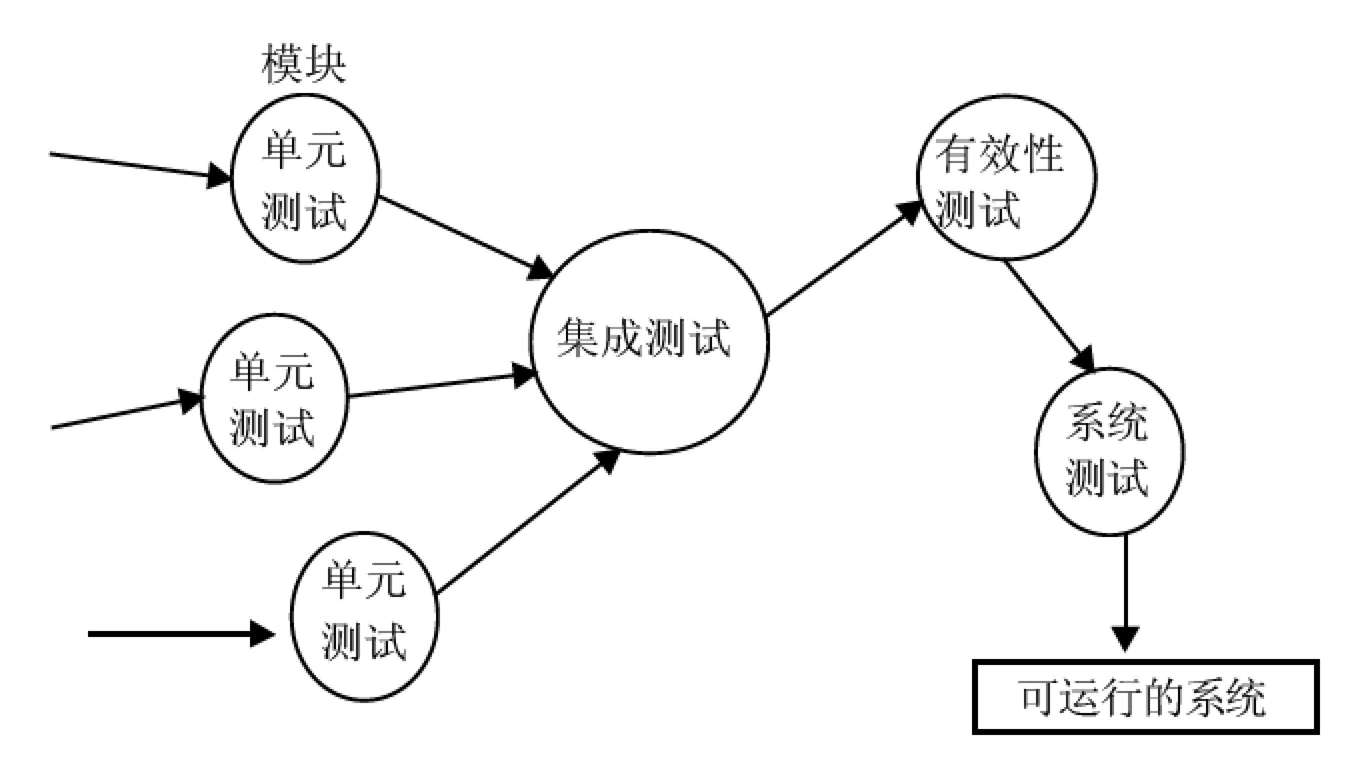


图 11-2　小系统测试实现的序列

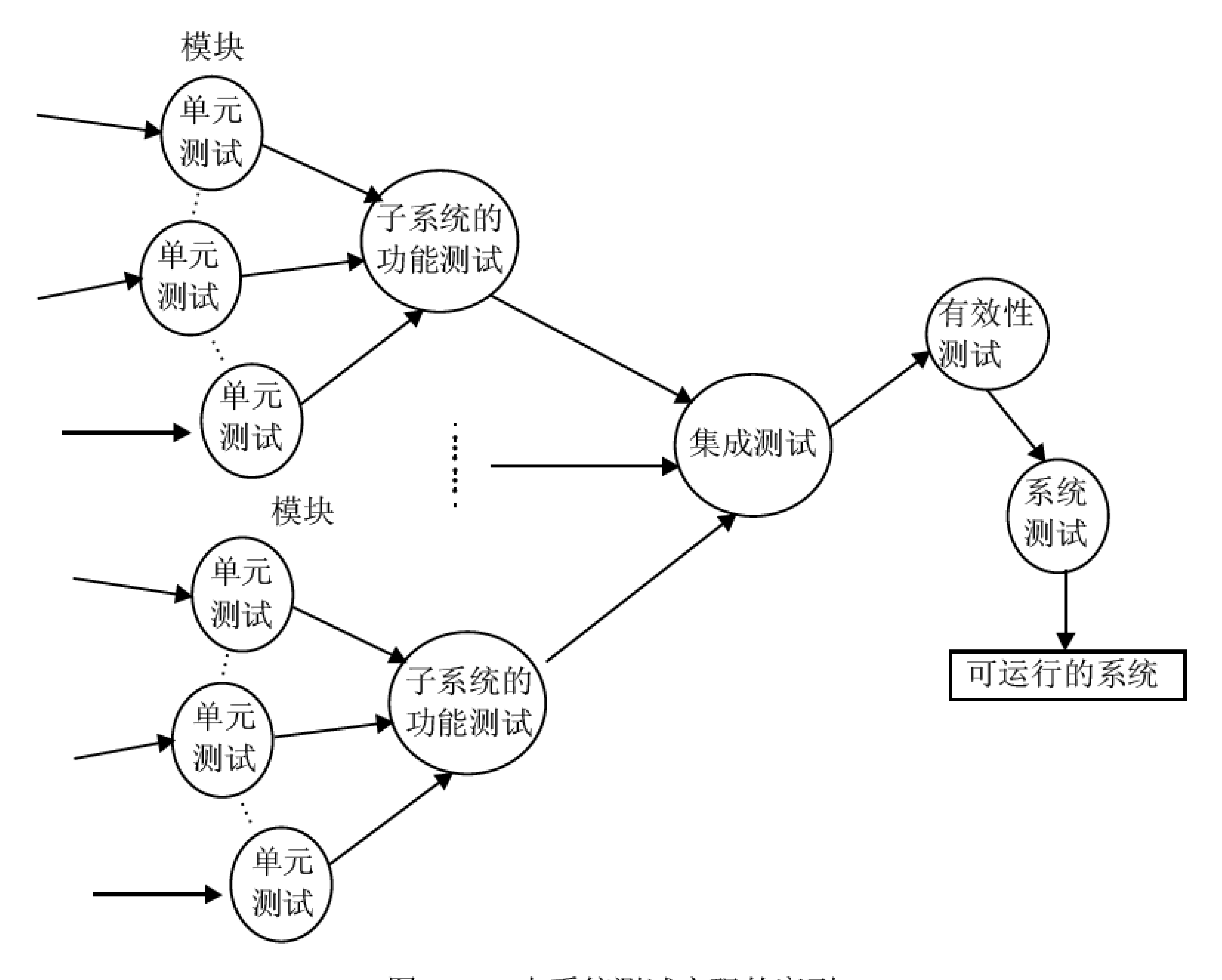


图 11-3　大系统测试实现的序列

11.3　系统测试的结果分析

对系统测试的结果分析如下。

1. 反应时间的性能测试

反应时间的性能测试如表 11-1 所示。

表 11-1　反应时间的性能测试

功能或事件	期望的反应时间	实际反应时间平均值（至少 3 次）	上次或上版本实际反应时间平均值（至少 3 次）
1			
2			
3			
…			
n			

2. CPU、Memory 的性能测试

CPU、Memory 性能测试的内容如下：

- 客户端情况。
- 应用服务器情况。
- 数据库服务器情况。

CPU、Memory 性能测试如表 11-2 所示。

表 11-2　CPU、Memory 性能测试

输入/动作	输出/响应	能否正常运行
10 个用户操作		
20 个用户操作		
40 个用户操作		
60 个用户操作		
80 个用户操作		
…		

3. 可靠性测试

可靠性测试通常使用 MTBF（Mean Time Between Failures）和 MTTR（Mean Time To Repairs）两个指标来衡量。

可靠性测试如表 11-3 所示。

表 11-3　可靠性测试

连续运行时间	建议 120 小时
故障发生的时刻	故障描述
…	…
统计分析	
功能 1 无故障运行的平均时间间隔	（CPU 小时）
功能 1 无故障运行的最小时间间隔	（CPU 小时）

（续表）

功能 1 无故障运行的最大时间间隔	（CPU 小时）
…	…
功能 n 无故障运行的平均时间间隔	（CPU 小时）
功能 n 无故障运行的最小时间间隔	（CPU 小时）
功能 n 无故障运行的最大时间间隔	（CPU 小时）

4. 网络性能测试

系统性能测试是对网络性能的结果分析，如网络流量、每秒采样数、网络延迟等。

5. 安全性测试

系统的安全性测试是检测系统的安全机制、保密措施是否完善，主要用于检验系统的防范能力。

6. 强度测试

强度测试（负载/压力测试）是对系统在异常情况下的承受能力的测试，是检查系统在极限状态运行时，性能下降的幅度是否在允许的范围内，主要从响应时间、处理速度、吞吐量、处理精度等方面来检测。测试需要验证系统能否在同一时间响应大量的用户、在用户传送大量数据的时候能否响应。

7. 安装测试

安装测试就是为了检测在安装过程中是否有误、是否容易操作等，主要监测系统的每一个部分是否齐全、硬配置是否合理、安装中需要产生的文件和数据库是否已产生、其内容是否正确等。

8. 恢复测试

恢复测试用于监测系统的容错能力。检测途径是采用各种方法使系统出现故障，检验系统是否能按照要求从故障中恢复过来，并在约定的时间内开始处理，且不对系统造成任何伤害。

11.4 系统测试的文档资料

系统测试阶段产生的文档资料如下：

- 系统测试报告书。
- 系统测试大纲。
- 测试目标。
- 测试内容。
- 程序测试（对每个程序）：包括程序测试的内容、程序测试的结果。
- 功能测试（对每个功能）：包括功能测试的内容、功能测试的结果。
- 子系统功能测试（对每个子系统）：包括子系统测试的内容、子系统测试的结果。
- 系统测试：包括系统测试的内容、系统测试的结果。
- 测试结果的评价，包括对程序的测试评价、对功能的测试评价、对子系统的测试评价、对系统的测试评价。

- 结论。
- 测试人员名单。
- 附录。
- 系统使用说明书草案。
- 系统维护手册草案。

第12章　验收测试技术

在完成系统测试阶段的工作后，进行验收测试。验收测试实际上是在产品阶段上进行测试，即要求被测试的软件系统和硬件系统与生产出来的产品的情况相同。

验收测试是每个系统的测试活动中所必须进行的。新建系统产品通过验收测试的工作后才能最终结束。

验收测试（Acceptance Testing）是以用户为主的测试。一般情况下，在软件系统测试结束以及软件配置审查之后开始，验收测试应由用户、测试人员、软件开发人员和质量保证人员一起参与，验证软件系统的功能、性能及其他特性是否与用户的要求一致。本章重点讨论以下内容：

- 验收测试的先决条件。
- 验收测试的目的。
- 验收测试的内容。

12.1　验收测试的先决条件

验收测试最基本的先决条件如下：

- 已通过测试评审。
- 新系统已通过试运行工作。
- 合同附件规定的各类文档齐全。
- 被测的系统应该是很稳定的，是要符合技术文档和标准规定的。
- 验收测试的时间是在系统被封版后进行的。

12.2　验收测试的目的

验收测试的目的主要包括如下几点：

- 新建系统产品是否按照需求开发的，体验该产品是否能够满足使用要求。
- 有没有达到原设计水平？完成的功能怎样？
- 对照合同的需求进行验收测试，是否符合双方达成的共识。
- 新建系统是否符合用户的需求，是否达到预期目的。
- 新建系统产品的可靠性和可维护性好不好。
- 新建系统产品通过运行表明对业务处理的能力。
- 新建系统产品对用户操作的容错能力。
- 新建系统产品对发生故障的恢复能力。
- 承建单位向业主单位提交的有关技术资料是否齐全。

12.3 验收测试的内容

新建系统产品已通过单元测试、功能测试、网络测试、软件安装测试、性能测试、集成测试、系统测试的测试阶段，在验收测试这一阶段的主要内容如下：

（1）对功能测试、网络测试、软件安装测试、性能测试、集成测试、系统测试的测试用例进行回归测试。

（2）验收测试组依据系统设计说明书的内容、系统使用说明书、系统维护手册，在新建系统产品中演示一遍，捕捉不足之处。需要做到如下几点：

- 新建系统产品是否运行正常，并达到预定的目标。
- 各个子系统是否运行正常，并达到预定的目标。
- 各个功能模块是否运行正常，并达到预定的目标。
- 按照系统使用说明书上介绍的方法去做能否实现。
- 按照系统维护手册上介绍的方法去做能否实现。

（3）测试文档验收，文档是否齐全、可信、符合标准。

（4）测试评估，从总体上对测试的质量进行评估。

（5）测试建议，指出本次测试工作的不足和需要在以后工作中改进的地方。

（6）文档测试，文档测试的主要内容如下：

- 将文档同程序相比较，看是否具有不相符的情况。
- 检查文档的截图是否与程序一致。
- 检查文档是否有错字或不符合语法规范的地方。
- 程序的帮助文档要说明准确、通俗易懂、不用专业术语且操作步骤要符合程序的要求。
- 文档要图文并茂、易于理解。
- 对文档要进行完整性、正确性、一致性、易理解性、易浏览性、版本统一性校验。

测试过程中涉及到的文档主要包括以下内容：

- 测试任务说明书。
- 测试计划说明书。
- 测试用例说明书。
- 测试报告说明书。
- 测试总结说明书。
- 测试验收说明书。
- 缺陷跟踪报告说明书。

第13章 Web测试技术

网站的 Web 测试与一般应用系统的软件测试不同，链接是 Web 应用系统的一个主要特征，不但需要检查和验证是否按照设计的要求运行，而且还要测试系统在不同用户的浏览器端的显示是否合适。重要的是，还要从最终用户的角度进行安全性和可用性测试。本章重点讨论以下内容：

- Web 的功能测试。
- Web 的性能测试（包括负载/压力测试）。
- Web 的用户界面测试。
- Web 的兼容性测试。
- Web 的安全性测试。
- Web 的接口测试。

13.1 Web 的功能测试

1. Web 的功能测试主要讨论的内容

Web 的功能测试主要讨论链接测试、表单测试、数据校验测试、Cookies 测试、数据库测试、应用程序特定的功能需求测试、权限测试。

（1）链接测试

链接是 Web 应用系统的一个主要特征，它是用户用于连接地址不明确页面的主要手段。

链接测试的重点如下：

- 测试所有链接是否按照指示的那样确实链接到了该页面。
- 测试所链接的页面是否存在。
- Web 应用系统上没有孤立的页面（所谓孤立页面是指没有链接指向该页面，只有知道正确的 URL 地址才能访问）。

链接测试一般是在集成测试阶段完成。

（2）表单测试

用户是通过表单提交信息链接进行测试的，表单应符合用户要求。在正常工作时需要确保以下问题：

- 如果使用表单来进行在线注册，需要确保“提交”按钮能正常工作，当注册完成后应返回注册成功的消息。
- 如果使用表单收集配送信息，应确保程序能够正确处理这些数据。
- 测试用户使用表单进行用户注册、登录、信息提交等操作时，能确保其完整性。

（3）数据校验测试

根据业务规则需要对用户输入进行校验，需要保证这些校验功能正常工作。

（4）Cookies 测试

Cookies 通常用来存储用户信息和用户在某应用系统的操作，当一个用户使用 Cookies 访问了某一个应用系统时，Web 服务器将发送关于用户的信息，把该信息以 Cookies 的形式存储在客户端计算机上，这可用来创建动态和自定义页面或者存储登录等信息。Cookies 测试的主要内容如下：

- 检查 Cookies 是否能正常工作。
- 是否按预定的时间进行保存，刷新对 Cookies 有什么影响。
- 在 Cookies 中保存注册信息，是否能对注册信息进行加密。
- 使用 Cookies 统计次数，需要验证次数是否累计正确。

（5）数据库测试

数据库为 Web 应用系统的管理、运行、查询和实现用户对数据存储的请求等提供空间。在 Web 应用中，最常用的数据库类型是关系型数据库，可以使用 SQL 对信息进行处理。

在使用了数据库的 Web 应用系统中，将重点测试可能发生的数据一致性错误和输出错误。数据一致性错误主要是由于用户提交的表单信息不正确而造成的，而输出错误主要是由于网络速度或程序设计问题等引起的，针对这两种情况，可分别进行测试。

（6）应用程序特定的功能需求测试

测试人员需要对应用程序特定的功能需求进行验证。可进行的所有操作是：下订单、更改订单、取消订单、核对订单状态、在货物发送之前更改送货信息、在线支付等。这也是用户使用网站的原因。

（7）权限测试

Web 系统的权限可分为功能权限、数据权限、操作权限、权限变化。

- 功能权限：指定用户可以使用哪些功能，不能使用哪些功能。
- 数据权限：指定用户可以处理哪些数据，不可以处理哪些数据，可以合并到功能测试。
- 操作权限：在逻辑关系上操作前后顺序、数据处理情况，可以合并到功能测试。
- 权限变化：可以合并到功能测试。

进行 Web 系统的权限测试时，应重点注意如下内容：

- 功能权限是否存在。
- 功能权限是否正确。
- 数据权限是否存在。
- 数据权限是否正确。
- 操作权限是否存在。
- 操作权限是否正确。
- 引起权限变化的功能列表是否存在。

- 是功能权限变化还是数据权限变化，或两者兼有。
- 权限变化是否正确。

2. Web 功能测试具体的测试内容

Web 功能测试的具体内容如下：

（1）Web 数据初始化是否执行。

（2）Web 数据初始化是否正确。

（3）Web 数据处理功能是否执行。

（4）Web 数据处理功能是否正确。

（5）Web 数据保存是否执行。

（6）Web 数据保存是否正确。

（7）Web 数据是否对其他功能有影响。

（8）Web 数据如果影响其他功能，系统能否作出正确的反应。

（9）对 Web 模块的所有功能进行测试，如某一功能模块具有最基本的（增加、修改、查询、删除）功能，则需要进行以下测试：

- 增加——增加——增加（连续增加测试）。
- 增加——删除。
- 增加——删除——增加。
- 增加——修改——删除。
- 修改——修改——修改（连续修改测试）。
- 修改——增加。
- 修改——删除。
- 修改——删除——增加。
- 删除——删除——删除（连续删除测试）。

13.2 Web 的性能测试

Web 的性能测试主要用于考察 Web 系统在不同负载条件下网站的行为。Web 的性能测试主要讨论：基准性能测试、负载测试、稳定性测试、压力测试等。

1. 基准性能测试

基准性能测试主要考察 Web 系统的响应时间、资源占用情况等。

- 响应时间：Web 系统的响应时间等于连接建立时间+响应时间。连接速度不能慢，若超过用户耐心等待的时间，则用户就会离开，要求希望不超过 5 秒钟。
- 资源占用：希望网站在正常响应用户请求的前提下，系统资源（CPU、内存、磁盘等）占用尽可能少。

2. 负载测试

负载测试主要是考察 Web 系统承受访问量的行为。

为了测试 Web 系统在某一负载级别上的性能，以保证 Web 系统在需求范围内能正常工作，负载测试的重点如下：

- 某个时刻同时访问 Web 系统的用户数量，也可以是在线数据处理的数量。
- Web 应用系统能允许多少个用户同时在线。
- 超过 Web 应用系统能允许多少个用户同时在线数量，会出现什么现象：①系统拒绝新的访问请求？② 系统殃及大批现有连接？③ 网站死机瘫痪？
- Web 应用系统能否处理大量用户对同一个页面的请求。

3. 稳定性测试

稳定性测试主要考察 Web 系统承受长时间运行的行为。

4. 压力测试

压力测试主要考察 Web 系统应用服务器承受用户请求量、并发请求量的行为。压力测试一般包含如下步骤：

01 确定接受请求并完成响应的最大允许的延时。

02 估计 Web 应用程序的最大并发用户数量。

03 模拟用户请求，以一个比较小的负载开始，逐渐增加模拟用户的数量，直到 Web 应用程序的相应延时超过最大延时。

压力测试的工具应提供以下的功能，以便进行测试：

- 发送 GET 和 POST 请求。
- “记录”从浏览器发送的 GET 和 POST 请求。
- 获取和发送 Cookies。
- 多线程。
- 模拟用户延迟。
- 记录性能数据。
- 控制带宽。

压力测试的区域包括表单、登录和其他信息传输页面等。压力测试应重点关注以下情况：

- 瞬间访问高峰。
- 每个用户传送大量数据。
- 长时间的使用。

13.3 Web 的用户界面测试

1. Web 的用户界面

Web 的用户界面主要讨论页面、页面元素和容错性。

（1）页面

页面应注意如下内容：

- 页面清单是否完整（是否已经将所需要的页面全部列出来了）？
- 页面是否显示（在不同分辨率下页面是否存在，在不同浏览器版本中页面是否显示）？
- 页面在窗口中的显示是否正确、美观（在调整浏览器窗口大小时，屏幕刷新是否正确）？
- 页面特殊效果（如特殊字体效果、动画效果）是否显示？
- 页面特殊效果显示是否正确？

（2）页面元素

页面元素应注意如下内容：

- 为实现 Web 的功能需要列出按钮、单选按钮、复选框、列表框、超链接、输入框等。
- 页面元素的文字、图形、签章是否显示正确？
- 页面元素的按钮、列表框、复选框、输入框、超链接等外形、摆放位置是否美观？
- 页面元素基本功能文字特效、动画特效、按钮、超链接是否实现？
- 页面元素是否存在容错性？

（3）容错性

容错性应注意如下内容：

- 为空、非空。
- 惟一性。
- 字长、格式。
- 数字、邮政编码、金额、电话、电子邮件、ID 号、密码。
- 日期、时间。
- 特殊字符（对数据库）、英文单/双引号、“&”符号。
- 页面元素的容错性是否正确？

2. Web 用户界面测试的内容

进行 Web 用户界面测试时，主要讨论的内容有：导航测试、图形测试、内容测试、表格测试、整体界面测试。

（1）导航测试

导航是 Web 应用系统描述用户在一个页面内操作的方式。导航用在不同的用户接口控制之间（如按钮、对话框、列表和窗口等）或不同的连接页面之间。

Web 应用系统的用户趋向于目的驱动，很快地扫描一个 Web 应用系统，看是否有满足自己需要的信息，如果没有，就会很快离开。很少有用户愿意花时间去熟悉 Web 应用系统的结构，因此，Web 应用系统导航帮助要尽可能地准确。在一个页面上放太多的信息往往会起到与预期相反的效果。

（2）图形测试

在 Web 应用系统中，适当的图片和动画既能起到广告宣传的作用，又能起到美化页面的功能。一个 Web 应用系统的图形可以包括图片、动画、边框、颜色、字体、背景、按钮等。图形测试的内容如下：

- 要确保图形有明确的用途，图片或动画不要胡乱地堆在一起，以免浪费传输时间。Web 应用系统的图片尺寸要尽量地小，并且要能清楚地说明某件事情，一般都链接到某个具体的页面。
- 验证所有页面字体的风格是否一致。
- 背景颜色应该与字体颜色和前景颜色相搭配。
- 图片的大小和质量也是一个很重要的因素，一般采用 JPG 或 GIF 压缩，最好能使图片的大小减小到 30kB 以下。
- 文字回绕是否正确。

（3）内容测试

内容测试用来检验 Web 应用系统提供信息的正确性、准确性和相关性。

- 信息的正确性是指信息是可靠的还是误传的。
- 信息的准确性是指是否有语法或拼写错误。
- 信息的相关性是指是否在当前页面中可以找到与当前浏览信息相关的信息列表或入口，也就是一般 Web 站点中的所谓“相关文章列表”。

（4）表格测试

表格测试需要验证表格是否设置正确。

（5）整体界面测试

整体界面是指整个 Web 应用系统的页面结构设计，是给用户的一个整体感。整体界面要求舒适，整个 Web 应用系统的设计风格应一致。

13.4 Web 的兼容性测试

Web 的兼容性包括操作系统兼容和应用软件兼容，可能还包括硬件兼容。

Web 的兼容性测试主要讨论：系统平台测试、浏览器测试、分辨率测试。

1. 系统平台测试

最常见的系统平台有 Windows、Unix、Macintosh、Linux 操作系统等。同一个应用可能在某些操作系统下能正常运行，但在另外的操作系统下不能运行，因此，Web 系统平台需要在各种操作系统下对 Web 系统进行系统平台兼容性测试。

2. 浏览器测试

浏览器是 Web 客户端最核心的构件，不同厂商的浏览器对 Java、HTML 规格有不同的支持，

框架和层次结构风格在不同的浏览器中也有不同的显示，浏览器需要测试兼容性。

3. 分辨率测试

页面版式在 640×400、600×800 或 1024×768 的分辨率模式下是否显示正常?字体是否太小或者是太大，以至于无法浏览?文本和图片是否对齐?

13.5　Web 的安全性测试

Web 的安全性测试主要讨论：目录设置、SSL、登录、日志文件。

1. 目录设置测试

Web 安全的第一步就是正确设置目录。每个目录下应该有 index.html 或 main.html 页面，这样就不会显示该目录下的所有内容。

2. SSL 测试

很多站点都使用 SSL 进行安全传送，使用 SSL 时需要确定是否有相应的替代页面（适用于 3.0 以下版本的浏览器，这些浏览器不支持 SSL）。

3. 登录测试

用户访问信息资料公开的站点时不需要进行登录，访问信息资料不公开的站点、商业站点时需要进行登录。进行登录操作，对于用户而言是不方便的。

有些站点需要用户进行登录，以验证他们的身份，并通过验证系统阻止非法的用户登录。登录主要测试如下内容：

- 用户登录是否有次数限制?
- 是否限制从某些 IP 地址登录?
- 如果允许登录失败的次数为 3，在第三次登录时输入正确的用户名和口令，能通过验证吗?
- 口令选择有规则限制吗?

4. 日志文件测试

在后台需要注意验证服务器日志工作是否正常。日志文件主要测试如下内容：

- 日志是否记录了所有的事务处理?
- 是否记录失败的注册?
- 是否记录被盗信用卡的使用?
- 是否在每次事务完成的时候都进行保存?
- 是否记录 IP 地址?
- 是否记录用户名?

13.6　Web 的接口测试

在很多情况下，Web 站点不是孤立的，可能会与外部服务器通信，并请求数据、验证数据或

提交订单。

Web 的接口测试主要讨论：服务器接口、外部接口。

1. 服务器接口测试

服务器接口测试主要对浏览器与服务器的接口进行测试。测试人员提交事务，然后查看服务器记录，并验证是否在浏览器上看到的正好是服务器上发生的。

2. 外部接口测试

有些 Web 系统存在外部接口，测试的时候，需要使用 Web 接口发送一些事务数据，分别对有效信用卡、无效信用卡和被盗信用卡进行验证。

第14章　自动化测试技术

为了节省人力、时间或硬件资源和提高测试效率，可使用自动化测试技术。本章重点讨论以下内容：

- 自动化测试概述。
- 自动化测试技术。
- 自动化测试级别。
- 自动化测试框架。
- 自动化测试工具。

14.1　自动化测试概述

1．自动化测试的概念

自动化的概念是人们在工业生产的过程中，为了提高工作效率，不断地对操作方法、技术或者工具进行改进，从而减少人的手工劳动，节省时间和成本。自动化测试是把以人为驱动的测试行为转化为机器执行的一种过程。通常，在设计测试用例并通过评审之后，由测试人员根据测试用例中描述的规程一步步执行测试，将得到的实际结果与期望结果进行比较。在此过程中，为了节省人力、时间或硬件资源和提高测试效率，便引入了自动化测试的概念，但是，自动化测试不能完全代替手工测试、立即降低测试投入，提高测试效率、保证100%的测试覆盖率。

自动化测试是软件行业测试技术，Bret Pettichord在《自动化测试的7个步骤》一书中讲到："我们对自动化测试充满了希望，然而，自动化测试却经常带给我们沮丧和失望；虽然，自动化测试可以把我们从困难的环境中解放出来，但在实施自动化测试解决问题的同时，又带来同样多的问题"。自动化测试节省人力、节省时间，得到的数据更精确些，而且操作的可重复性和Bug的可重现性更强一些，而软件行业的测试有节约成本、提高效率的需求，所以自动化测试需要考虑到成本的问题。若要理解自动化测试，需要从自动化测试的前提和成本两个方面进行考虑。

（1）自动化测试的前提

实施自动化测试之前需要对软件开发过程进行分析，以观察其是否适合使用自动化测试。通常需要同时满足以下条件。

- 产品本身特征具有长期可维护性。
- 软件需求变动不频繁：项目中的某些模块相对稳定，而某些模块需求变动性很大时，便可对相对稳定的模块进行自动化测试，而变动较大的仍然使用手工测试。
- 项目周期足够长：由于自动化测试需求的确定、自动化测试框架的设计、测试脚本的编写与调试均需要较长的时间来完成。如果项目的周期比较短，没有足够的时间去支持这样一

个过程，那么不适合使用自动化测试。

- 产品本身非紧迫的大项目。
- 产品结构相对复杂。
- 资源投入相对充裕。
- 在手工测试无法完成、需要投入大量时间与人力时，也需要考虑引入自动化测试，如性能测试、配置测试、大数据量输入测试等。

（2）成本

有的人说：“从管理的角度来说，100%的自动化目标只是一个从理论上可能达到的，但是实际上达到 100%的自动化的代价是十分昂贵的。一个 40%～60%的自动化利用程度已经是非常高了，达到这个级别以上，需要增加测试相关的维护成本”。成本包括以下内容：

- 实现成本。
- 人力成本。
- 新技术的风险。
- 大量的手工劳动。

2. 自动化测试的过程

自动化测试无非是利用自动化测试工具，经过对测试需求的分析，设计出自动化测试用例，从而搭建自动化测试的框架、设计与编写自动化脚本、测试脚本的正确性，同样需要经历需求分析、制定测试计划、设计测试用例、执行测试、撰写测试报告、消除软件缺陷、评估测试结果等，过程如图 14-1 所示。

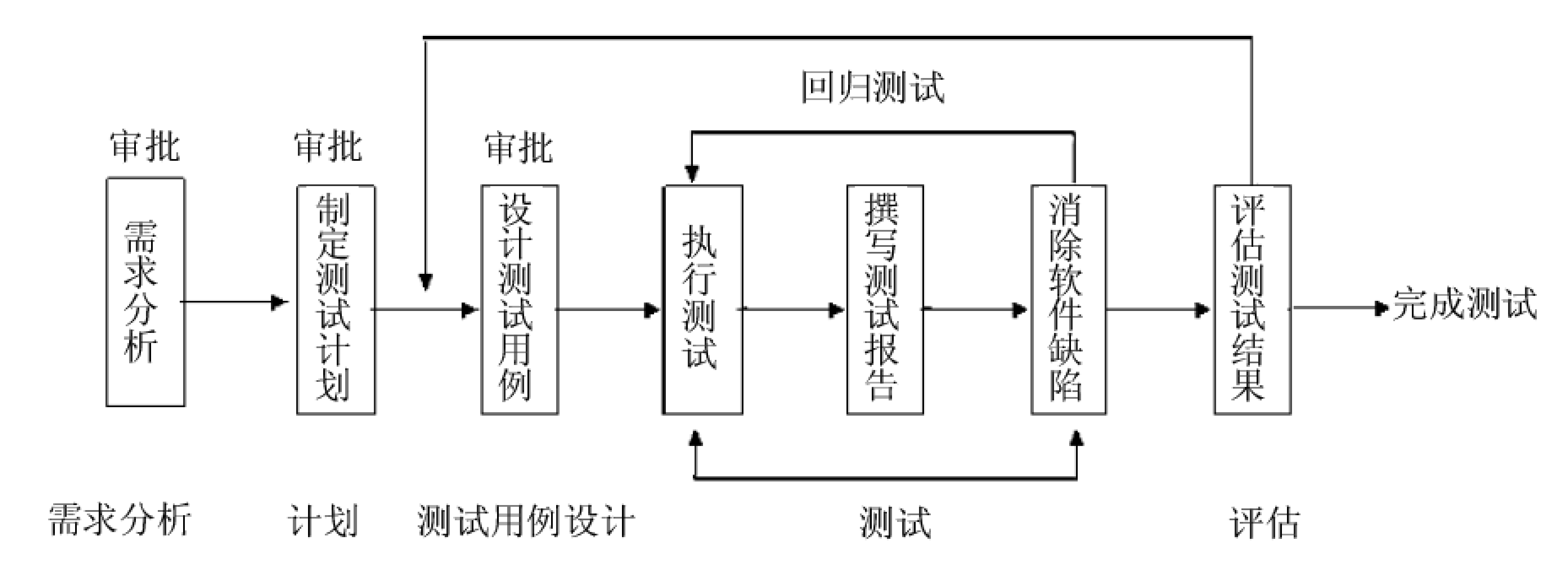


图 14-1 自动化测试过程

（1）需求分析

当测试项目满足了自动化的前提条件，并确定在该项目中需要使用自动化测试时，便可开始进行自动化测试需求分析，可落实到《测试需求说明书》。

（2）制定测试计划

此过程需要确定自动化测试的范围以及相应的测试用例、测试数据，并形成详细的文档，以便于自动化测试框架的建立，包括定义测试活动模型（确定测试所使用的测试技术）、定义测试体系结构、完成测试程序的定义与映射（建立测试程序与测试需求之间的联系）、自动/手动测试映

射（确定哪些测试使用自动测试）以及测试数据映射。决定如何测试、划分测试阶段、类型以及测试方法等。

（3）设计测试用例

自动化测试的框架可确定需要调用哪些文件、结构、调用过程、文件结构如何划分等内容。 自动化测试框架的典型要素如下。

- 公用的对象：不同的测试用例会有一些相同的对象被重复使用，这些公用的对象可被抽取出来，在编写脚本时随时调用。
- 公用的环境。
- 公用的方法。

（4）执行测试

以上准备工作均做好后，就可以执行测试。

（5）撰写测试报告

执行测试后，编写测试用例或开发测试脚本，并将其文档化。

（6）消除软件缺陷

调试测试（针对自动化测试脚本），并修改软件缺陷。

（7）评估测试结果

测试执行结束后，需要对测试结果进行比较、分析以及结果验证，得出测试报告。其中总结性报告是提供给被测方中高层管理者及客户的，而详细报告作为反馈文档提供给开发小组成员。评估过程如图 14-2 所示。

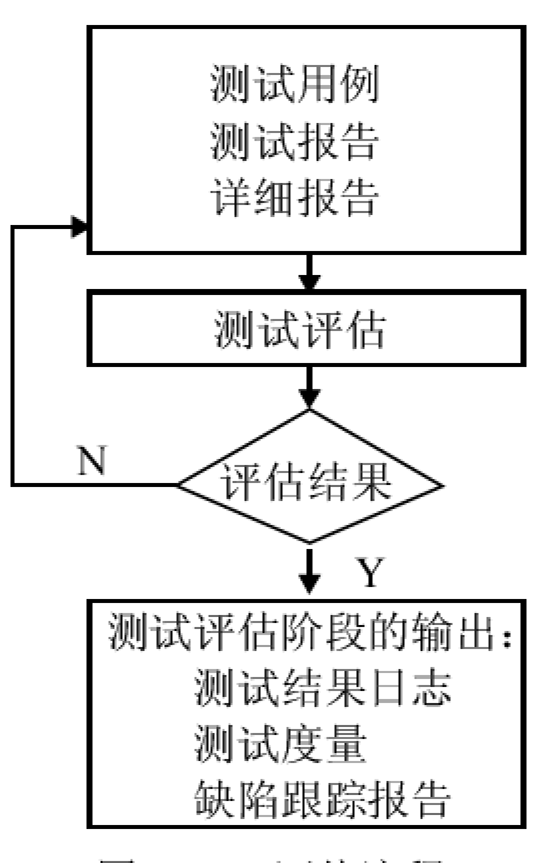


图 14-2　评估流程

这个阶段由测试设计工程师与测试工程师共同参与。在构建好的待测系统上，使用测试用例脚本执行测试数据，其中测试数据是被设计用于测试该应用程序各种特征的。可以使用 Excel、Word、ClearQuest 等工具得到测试结果日志、测试度量、缺陷报告及测试评估总结等。

测试执行结束后，需要对测试结果进行比较、分析以及结果验证，得出测试报告（包括总结性报告和详细报告）。

3. 自动化测试的特点

自动化测试具有以下优点：

- 能执行更多、更频繁的测试，使某些测试任务的执行比手动方式更高效，可以更快地将软件推向市场。
- 能执行一些手动测试困难或不可能做的测试。
- 更好地利用资源，利用整夜或周末空闲的设备执行自动化测试。
- 将任务自动化，让测试人员投入更多的精力设计出更多、更好的测试用例，提高测试准确性和测试人员的积极性。
- 自动测试具有一致性和可重复性的特点，而且测试更客观，提高了软件的信任度。

但自动化测试仍然存在着一定的局限性，具有以下缺点：

- 不能取代手工测试，不可能自动化所有的测试，如测试只是偶尔执行，或待测系统经常变动、不稳定，测试需要大量的人工参与时，就不适宜采用自动测试。
- 自动测试工具本身不具有想象力，只是按命令执行。而手工测试时测试执行者可以在测试中判断测试输出是否正确，以便改进测试，还可以处理意外事件。
- 自动测试对测试质量的依赖性较大，在确保测试质量的前提下，实施自动化测试才有意义。
- 自动测试在刚开始执行时，工作效率并不一定高于手动测试，只有当整个自动测试系统成熟，且测试工程师熟练掌握测试工具后，工作效率才会随着测试执行次数的增加而提高。
- 自动测试的成本可能高于手工测试。自动测试的成本大致由以下几个部分组成：自动测试开发成本、自动测试运行成本、自动测试维护成本和其他相关任务带来的成本、软件的修改带来测试脚本部分或全部修改，就会增加测试维护的开销。

14.2 自动化测试技术

软件自动化测试技术主要有：录制/回放、脚本技术、数据驱动、关键字驱动、业务驱动等。

1. 录制/回放

所谓的“录制/回放”就是先由手工完成一遍需要测试的流程，由工具记录下这个流程期间客户端和服务器端之间的通信过程，记录下用户和应用程序交互时的击键和鼠标的移动，形成一个脚本，然后可以在测试执行期间“回放”。

在这种模式下数据和脚本混在一起，几乎一个测试用例对应一个脚本，维护成本很高，而且即使界面的简单变化也需要重新录制，脚本可重复使用的效率低。

2. 脚本技术

脚本是一组测试工具执行的指令集合，也是计算机程序的一种形式。脚本可以通过录制测试的操作产生，然后再做修改，这样可以减少脚本编程的工作量。

脚本技术分为线性脚本、结构化脚本、共享脚本、数据驱动脚本、关键字驱动脚本等。

- 线性脚本：线性脚本是录制手工执行的测试用例得到的脚本，这种脚本包含所有的击键、移动、输入数据等，所有录制的测试用例都可以得到完整的回放。
- 结构化脚本：结构化脚本类似于结构化程序设计，具有各种逻辑结构（顺序、分支、循环），而且具有函数调用功能。
- 共享脚本：共享脚本是指某个脚本可被多个测试用例使用，即脚本语言允许一个脚本调用另一个脚本。
- 数据驱动脚本：将测试输入存储在独立的数据文件中。
- 关键字驱动脚本：关键字驱动脚本是数据驱动脚本的逻辑扩张。

3. 数据驱动

数据驱动从数据文件读取输入数据，通过变量的参数化将测试数据传入测试脚本，不同的数据文件对应不同的测试用例。在这种模式下数据和脚本分离，脚本的利用率、可维护性大大提高，但受界面变化的影响仍然很大。

4. 关键字驱动

关键字驱动测试是数据驱动测试的一种改进类型，它将测试逻辑按照关键字进行分解，形成数据文件，关键字对应封装的业务逻辑，主要关键字包括三类：被操作对象（Item）、操作（Operation）和值（Value），利用面向对象的形式可将其表现为 Item.Operation(Value)。关键字驱动的主要思想是：脚本与数据分离、界面元素名与测试内部对象名分离、测试描述与具体实现细节分离。

目前，大多数测试工具处于数据驱动到关键字驱动之间的阶段，有些工具厂商已支持关键字驱动的版本。

5. 业务驱动

业务驱动可分为接入层业务驱动、业务层业务驱动、数据层业务驱动、性能驱动。驱动的过程如图 14-3 所示。

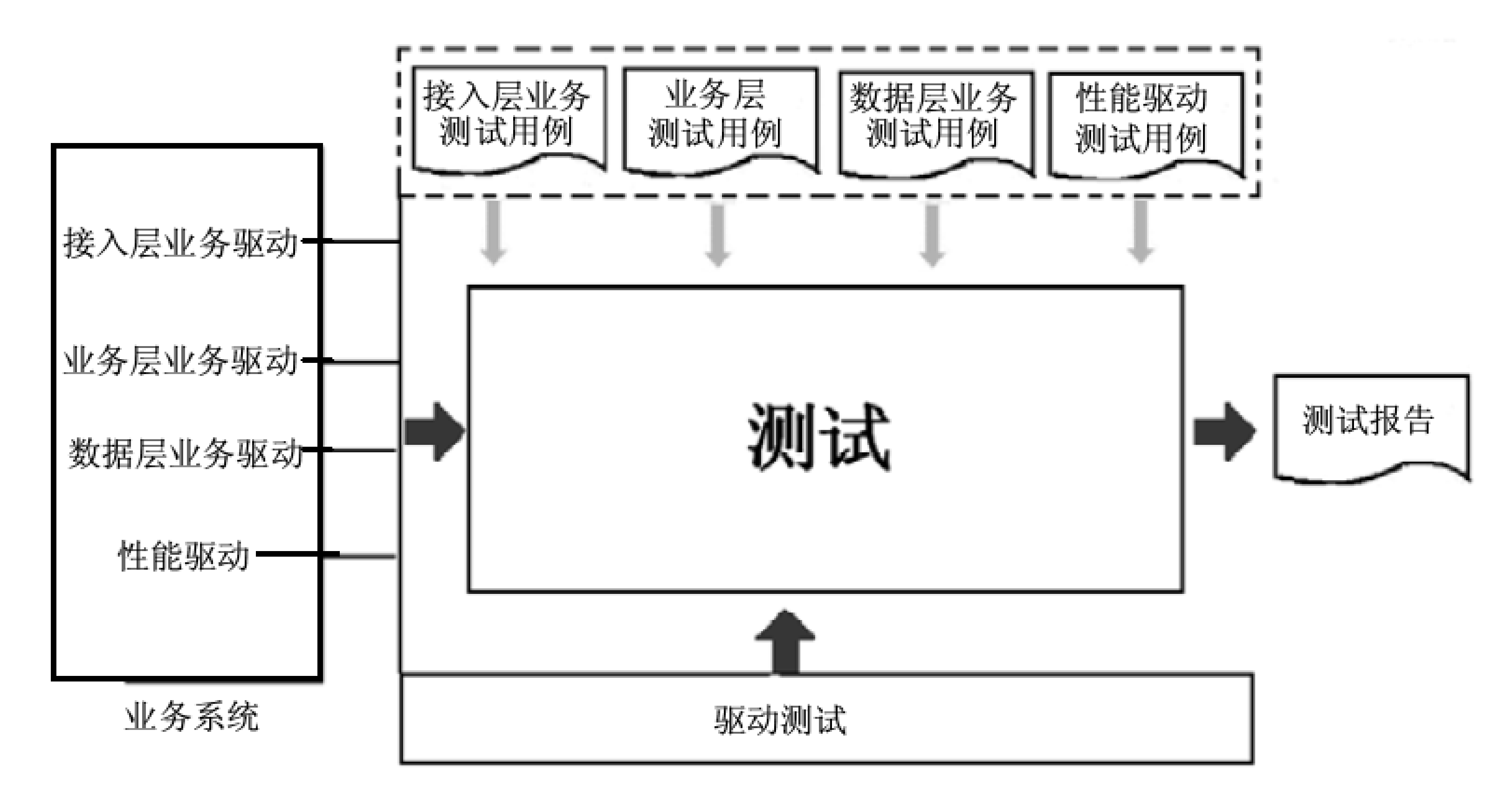


图 14-3　业务驱动的过程图

自动化测试技术的比较如表 14-1 所示。

表 14-1 自动化测试技术的比较

	录制回放	脚本技术	数据驱动	关键字驱动	业务驱动
可维护性	低	中	中	中	高
可靠性	低	高	高	高	高
效率	低	中	中	中	高
易用性	高	低	中	中	高
可移植性	低	低	中	中	中
可复用性	低	中	中	高	高
健壮性	低	高	高	高	高

14.3 自动化测试级别

自动化测试工具只是被看作是一种捕获、编辑、编程、数据驱动、使用动作词和回放的工具，被划分为 5 个级别。

1. 级别 1：捕获和回放

级别 1 是使用自动化测试的最低级别，在级别 1 上，可建立捕获和回放。

- 级别 1 的优点：自动化测试的脚本能够被自动生成，而不需要有任何的编程知识。
- 级别 1 的缺点：拥有大量的测试脚本，同时当需求和应用发生变化时相应的测试脚本也必须被重新录制。
- 级别 1 的用法：捕获和回放用于当测试的系统不会发生变化时，小规模的自动化测试。

2. 级别 2：捕获、编辑和回放

在级别 2 中，使用自动化测试工具来捕获想要测试的功能。建立捕获、编辑和回放，将测试脚本中的任何固定的测试数据，如名字、账号等，从测试脚本的代码中完全删除，并将它们转换成为变量。

- 级别 2 的优点：测试脚本开始变得更加完善和灵活，并且可以大大减少脚本的数量和维护的工作。
- 级别 2 的缺点：需要一定的编程知识，并且变更和维护几乎是不可能的。
- 级别 2 的用法：当进行回归测试时，被测试的应用有很小的变化，如仅仅是针对计算的代码变化，但是没有关于 GUI 界面的变化。

3. 级别 3：编程和回放

级别 3 是面对多个构建版本的有效使用测试自动化的第一个级别。需要在实际的投资开始显现之前确保团队和客户对项目的安全感。如果没有对测试自动化的工具适当地培训的话，测试人员将不具备到达这个级别的能力。

在自动化测试工具中的所有测试功能都必须被很好地理解，并且要掌握测试脚本语言的知识。

- 级别 3 的优点：确定了测试脚本的设计。测试人员要有适当的编码，使用与开发中相同的编码习惯是非常好的。能够在项目的早期就开始进行测试脚本的设计。

- 级别 3 的缺点：要求测试人员具有很好的软件技能，包括设计、开发等。
- 级别 3 的用法：大规模的测试套件被开发、执行和维护的专业自动化测试。级别 3 使您能够使用自动化测试并构建不同的回归测试（重用已有的自动化测试用例）。

4. 级别 4：数据驱动的测试

级别 4 是一个专业的测试级别，需要利用测试工具提供的所有的测试功能，它拥有一个强大的测试框架，这个测试框架是基于被测试系统的变化快速创建的一个测试脚本的测试功能库。维护的成本相对是比较低的。在测试中会使用到大量真实的数据。

- 级别 4 的优点：能够维护和使用良好的并且有效的模拟真实生活中数据的测试数据。
- 级别 4 的缺点：软件开发的技能是基础，并且需要访问相关的测试数据。
- 级别 4 的用法：大规模的测试套件被开发、执行和维护的专业自动化测试。级别 4 要求一些非常良好的测试数据。一个测试人员必须要花费一些时间来识别在哪里收集数据和收集哪些数据。

5. 级别 5：使用动作词的测试自动化

级别 5 是自动化测试的最高级别，主要的思想是将测试用例从测试工具中分离出来。级别 5 要求有一个具有高技能测试人员的团队，这些测试人员能够将测试工具非常深的层次知识与他们具备的较深的编程能力结合起来。这个团队负责在测试工具中生成并维护测试的功能性，能够使测试工具从外部的来源，如 Excel 表或者数据库中执行测试用例。在 Excel 表中创建测试用例时，放置使用包括被使用的特定动作词语的一些类型模板。执行的过程是从 Excel 表中读取测试用例，并将测试用例转换成为测试工具能够理解的形式，然后使用不同的测试功能来执行测试。

- 级别 5 的优点：测试用例的设计被从测试工具中分离了出来；允许快速测试用例的执行和基于用例的更好地估计。
- 级别 5 的缺点：需要一个具有工具技能和开发技能的测试团队，以提供并维护测试工程（框架）。
- 级别 5 的用法：专业的测试自动化能够使用 Excel 来生成实际的测试用例，当用例适当时需要做的工作也是非常简单的，可以集中时间来生成一个包含被需要的“对象映射”的测试用例（主流程）。通常多数的测试用例可以复制已有的测试用例，并对其进行必要的修改，通常这种修改是有限的。动作词语框架能够通过使用用例使紧密的并行测试用例的开发变为可能。

14.4 自动化测试框架

一个支持多平台的自动化测试框架模型 SAFS（Software Automation Framework Support）是一个开源的支持多平台的自动化测试框架，由 SAS Institute 的 Carl Nagle 开发。支持多平台的自动化测试框架模型的结构如图 14-4 所示。

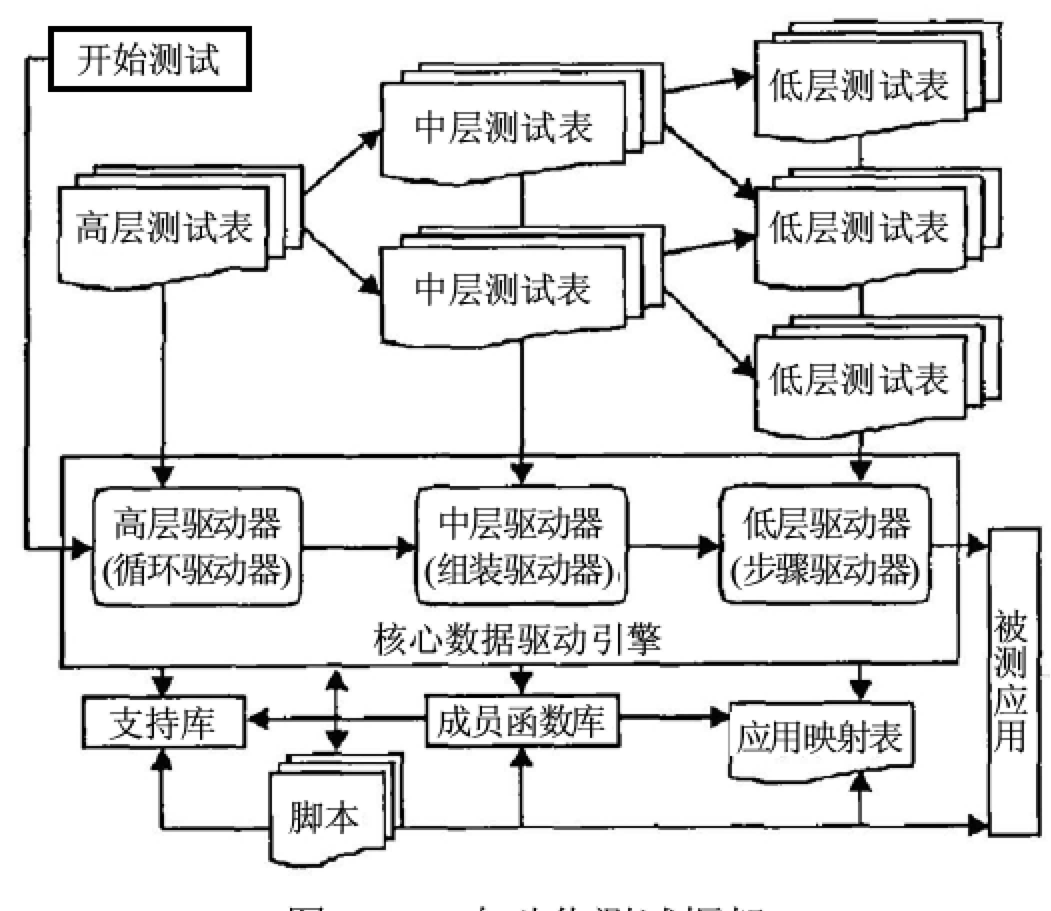


图 14-4 自动化测试框架

图 14-4 是由测试表、核心数据驱动引擎、成员函数库、支持库、应用映射表组成的。

1. 测试表

测试表（Test Tables）用于保存测试数据和关键字，分为高层测试表、中层测试表、低层测试表。其中，下层的测试表被上层的测试表所调用。

2. 核心数据驱动引擎

核心数据驱动引擎（Core Data Driven Engine）与测试表对应，分为高层驱动器（也叫循环驱动器）、中层驱动器（也叫组装驱动器）和低层驱动器（也叫步骤驱动器）。高层驱动器读取相应测试表的关键字逐级传递给下层的驱动器，最后由低层驱动器调用关键字库中的指令对应的组件函数来执行。

3. 成员函数库

成员函数库（Component Function）实现了用户对界面对象的各种操作指令，它在被测应用和自动化工具之间提供了一个隔离层。

4. 支持库

支持库（Support Libraries）是通用的程序和工具库，提供诸如数据库访问、字符串操作、文件访问、日志记录等基础性的支持功能。

5. 应用映射表

应用映射表（Application Map）是对应用中的对象定义一套命名规范，将这些实际对象的名字和自动化工具识别的对象名联系起来，形成映射表，从而使应用对象元素和测试对象名分离，提高了脚本的可维护性。

14.5 自动化测试工具

随着人们对软件测试工作的重视，涌现出了大量的软件测试自动化工具，为提高软件测试的效率和软件测试的质量提供了方法。

14.5.1 自动化测试工具的特征

自动化测试工具具有如下特征：

- 支持脚本化语言。
- 具有对程序界面中对象的识别能力。
- 支持函数的可重用性。
- 支持外部函数库。
- 具有错误处理能力。
- 具有调试器。
- 具有源代码管理功能。
- 抽象层能将程序界面中的对象实体映射成逻辑对象。
- 支持分布式测试。
- 支持数据驱动测试。
- 支持脚本的命令行方式。

14.5.2 自动化测试工具的分类

测试工具可以从不同的方面去分类。

1. 测试方法

按照测试方法进行分类时，自动化测试工具可以分为以下两类：

- 白盒测试工具。
- 黑盒测试工具。

2. 捕获错误用途

捕获错误顾名思义就是用于捕获软件错误或程序调试。按照捕获错误用途进行分类时，自动化测试工具可以分为以下几类。

- 一般用途：这里所说的一般用途，是指这个测试工具在进行测试时，可以适用于大部分的软件。
- GUI 自动化用途：目前大多数自动化测试软件除了提供在窗口界面中使用外，也有不少是针对浏览器接口开发的自动化测试工具。
- 专项用途：专项用途就是指某种专项测试的软件，如专用代码测试工具、白盒测试工具、黑盒测试工具、网络测试工具。
- 软件产品功能、性能测试用途：这类测试工具通过自动录制、检测和回放用户的应用操作，将被测系统的输出记录同预先给定的标准结果进行比较。

- 测试管理工具：测试管理工具用于对测试进行管理。
- 测试辅助工具：这些工具本身并不执行测试，例如它们可以生成测试数据、为测试提供数据准备等。

3. 语言

按照语言进行分类时，自动化测试工具可以分为以下类型：

- C/C++单元级测试工具。
- JUnit（是一个开放源代码的 Java 测试框架）。

4. 工具的功能特点

按照工具的功能特点进行分类时，自动化测试工具可以分为以下类型：

- 内存资源泄漏检查工具。
- 代码覆盖率检查工具。
- 代码性能检查工具。
- 软件纠错工具。
- 基于 Web 应用的测试工具。
- GUI 功能测试工具。
- 面向开发的单元测试工具。
- 负载和性能测试工具。
- 软件测试管理和其他工具。

14.5.3　自动化测试工具的常用类型

常用的软件测试工具主要有：负载压力测试、功能测试、白盒/黑盒测试、测试管理、测试辅助、GUI 接口自动化测试工具。

1. 负载压力测试

负载压力测试工具的种类相当多，主要有以下几种类型。

（1）LoadRunner

LoadRunner 是 Mercury Interactive 公司开发的一种预测系统行为和性能的负载测试工具，它可以通过模拟成千上万个用户和实施实时监测来确认和查找问题。对于有实力的大公司而言，这款软件可能比较适合，它的功能和 QALoad 相比不相上下。通过使用 LoadRunner，企业能够最大限度地缩短测试时间、优化性能和加速应用系统的发布周期。

LoadRunner 是一种具有较高规模适应性的自动负载测试工具，它能预测系统行为、优化性能，强调的是整个企业的系统。

LoadRunner 的特点如下：

- 支持的协议多且个别协议支持的版本比较高。
- 负载压力测试方案设置灵活。
- 丰富的资源监控。

- 报告可以导出到 Word、Excel 以及 HTML 格式。

（2）QACenter

QACenter 自动化测试系列工具是 Compuware 公司的产品，它能够帮助测试人员创建快速、可重用的测试过程。这些测试工具可以帮助管理测试过程、快速分析和调试程序，包括针对回归、强度、单元、并发、集成、移植、容量、负载测试、自动执行测试和产生测试结果文档等。

QACenter 主要包括：QARun、QALoad、QADirector、EcoTools 和 TESTBytes 等模块。

QACenter 的特点如下：

- 测试接口多。
- 可预测系统性能。
- 通过重复测试寻找瓶颈问题。
- 从控制中心管理全局负载测试。
- 可验证应用的扩展性。
- 快速创建仿真的负载测试。
- QALoad 不单单测试 Web 应用，还可以测试一些后台的东西，如 SQL Server 等。只要它支持的协议，都可以测试。
- 性能价格比较高。

（3）Benchmark Factory

Benchmark Factory 的特点如下：

- 测试服务器群集的性能。
- 实施基准测试。
- 生成高级脚本。

（4）E-Test Suite

E-Test Suite 是由 Empirix 公司开发的测试软件，是能够和被测试应用软件无缝结合的 Web 应用测试工具。工具包含 e-Tester、e-Load 和 e-Monitor，这三种工具分别对应功能测试、压力测试以及应用监控，每一部分功能相互独立，测试过程又可彼此协同。

（5）JMeter

JMeter 是一个专门为运行和服务器负载测试而设计、100%的纯 Java 桌面运行程序。原先它是为 Web/HTTP 测试而设计的，但是它已经扩展以支持各种各样的测试模块。它与 HTTP 和 SQL（使用 JDBC）的模块一起运行，可以用来测试静止或活动资料库中的服务器运行情况、模拟服务器或网络系统在重负载下的运行情况。它也提供了一个可替换的界面，用来定制数据显示、测试同步及测试的创建和执行。

（6）WAS

WAS 是 Microsoft 提供的免费的 Web 负载压力测试工具，应用广泛。WAS 可以通过一台或者多台客户机模拟大量用户的活动。WAS 支持身份验证、加密和 Cookies，也能够模拟各种浏览器和 Modem 速度。

（7）ACT

ACT 或称 MSACT，它是微软的 Visual Studio 和 Visual Studio.NET 自带的一套进行程序压力测试的工具。ACT 不但可以记录程序运行的详细数据参数，利用图表显示程序运行情况，而且安装和使用都比较简单，结果阅读也很方便，是一套较理想的测试工具。

（8）OpenSTA

OpenSTA（Open System Testing Architecture）的特点是可以模拟很多用户来访问需要测试的网站，它是一个功能强大、自定义设置功能完备的软件，但是，这些设置大部分需要通过 Script 来完成，因此在真正使用这个软件之前，必须学习好它的 Script 编写。如果需要完成很复杂的功能，Script 的要求还比较高。当然这也是它的优点，一些程序员不会在意编写 Script 的。

（9）PureLoad

PureLoad 是一个完全基于 Java 的测试工具，它的 Script 代码完全使用 XML，所以，编写 Script 很简单。它的测试包含文字和图形，并可以输出为 HTML 文件。由于是基于 Java 的软件，因此 PureLoad 可以通过 Java Beans API 来增强软件功能。

2. 功能测试

目前，市场上功能测试工具的种类相当多，主要有 WinRunner、Rational Robot、Functional Tester。

（1）WinRunner

WinRunner 是 Mercury Interactive 公司提供的一个企业级的功能检测工具。WinRunner 使功能测试得以自动化，从而保证了应用程序按照预定方式运行。它以测试脚本形式将业务的过程记录下来，并随着相应的应用程序的开发或更新来支持对脚本的改进。执行脚本及报告结果在整个的应用周期中可对脚本重复使用，用于检测应用程序是否能够达到预期的功能及正常运行，自动执行重复任务并优化测试工作，从而缩短测试时间。通过自动录制、检测和回防用户的应用操作提高测试效率。

（2）Rational Robot

Rational Robot 测试工具属于 Rational TestSuite 中的一员，对于 Visual Studio 6.0 编写的程序支持非常好，同时还支持 Java Applet、HTML、Oracle Forms、People Tools 应用程序。要支持 Delphi 程序的测试还必须下载插件。Rational Robot 的语法使用 Basic，它的语言使用 SQABasic。

（3）Functional Tester

Functional Tester 是 Robot 的 Java 实现版本，是在 Rational 被 IBM 收购后发布的。在 Java 的浪潮下，Robot 被移植到了 Eclipse 平台上，并完全支持 Java 和.NET。可以使用 VB.NET 和 Java 进行脚本的编写。

3. 白盒/黑盒测试工具

白盒测试工具主要有以下几种：

- Logiscope。
- PRQA。

- JUnit。
- DevPartner。
- Numega。
- Pure。

黑盒测试工具主要有：

- QACenter。
- SQATeamTest。
- Rational Visual Test。

4. 测试管理

TestDirector 是 MI 公司的测试管理工具，可以与 WinRunner、LoadRunner、QuickTestPro 进行集成。除了可以跟踪 Bug 外，还可以编写测试用例、管理测试进度等，是测试管理的首选软件。

TestManager 是 Rational TestSuite 中的一员，可以用来编写测试用例、生成 Datapool、生成报表、管理缺陷以及日志等。缺点是必须和其他组件一起使用，测试成本比较高。

Bugzilla 是一个产品缺陷的记录及跟踪工具，它能够建立一个完善的 Bug 跟踪体系，包括报告、查询并产生报表、处理解决等几个部分。它的主要特点为：基于 Web 方式，安装简单；有利于缺陷的清楚传达；系统灵活，可配置性很强；自动发送 Email。

Jira 是一个 Bug 管理工具，自带一个 Tomcat 4；同时有简单的工作流编辑，可用来定制流程；数据存储在 HSQL 数据引擎中，因此只要安装了 JDK 就可以使用了。相比较 Bugzilla 来说，具有不少自身的特点，不过可惜它并不是开源工具，具有 Lisence 限制。

5. 测试辅助

SmartDraw 用于绘制 UCML，进行负载压力测试需求分析。对压力测试前的工作很有帮助。

6. GUI 接口自动化测试工具

目前市场上有关 GUI 形式的自动化测试软件种类相当多，而且所支持的操作平台也越来越多。基本上 GUI 自动测试的原理就是以录制和播放（Record and Replay）为主要的操作方式。由于每一家开发公司所采用的开发技术不同，因此使用者所要学习的指令编写方式也大不相同。

自动化测试时，测试工具的选择很重要，目前还没有一个单一的测试工具能用来完成所有的测试需求。测试工具品种不一、功能性能各异，对自动测试工具的适当选择，很大程度上决定了该工具能否获得相应的投资回报。

测试工具可分为以下两类。

- 找错工具（fault-finding）：根据既定的测试标准寻找被测程序中的缺陷，包括静态分析工具（一些白盒测试工具，例如 Parasoft 的 JTest 含有该功能）、动态测试工具（市面众多的测试工具，如 Robot、WinRunner、QARun 等）。
- 测试支持工具：测试管理工具（如 TestManager、TestDirector 等）、其他支持工具（如 ClearCase、ClearQuest 等）。

自动化测试工具选型的参考性原则如下：

- 选择尽可能少的自动化产品覆盖尽可能多的平台，以降低产品投资和团队的学习成本。
- 测试流程管理自动化通常应该优先考虑，以满足为企业测试团队提供流程管理支持的需求。
- 在投资有限的情况下，性能测试自动化产品将优先于功能测试自动化被考虑。
- 在考虑产品性价比的同时，应充分关注产品的支持服务和售后服务的完善性。
- 尽量选择趋于主流的产品，以便通过行业间交流甚至网络等方式获得更为广泛的经验和支持。
- 应对测试自动化方案的可扩展性提出要求，以满足企业不断发展的技术和业务需求。

第15章　面向对象的测试技术

面向对象技术是一种软件开发技术，正逐渐被广泛使用。面向对象技术能产生更好的系统结构，更规范的编程风格，极大地优化了数据使用的安全性，提高了程序代码的重用，尽管面向对象技术的基本思想保证了软件应该有更高的质量，但实际情况却并非如此，因为无论采用什么样的编程技术，编程人员的错误都是不可避免的，而且由于面向对象技术开发的软件代码重用率高，更需要严格测试，从而避免错误的繁衍。

面向对象程序的结构不是传统的功能模块结构，抛弃了传统的开发模式，对每个开发阶段都有不同以往的要求和结果，已经不可能用功能细化的观点来检测面向对象分析和设计的结果，因此，传统的测试模型对面向对象软件已经不再适用。

本章重点讨论以下内容：

- 面向对象的测试概述。
- 面向对象的测试模型。
- 面向对象分析的测试。
- 面向对象设计的测试。
- 面向对象编程的测试。
- 面向对象的单元测试。
- 面向对象的集成测试。
- 面向对象的系统测试。
- 面向对象的测试用例设计。

15.1　面向对象的测试概述

面向对象 OO（Object Oriented）的基本思想是使用对象、类、继承、封装、消息、抽象等基本概念来进行程序设计。它是从现实世界中客观存在的事物（对象）出发来构造软件系统，并在系统构造中尽可能运用人类的自然思维方式，强调直接以问题域（现实世界）中的事物为中心来思考问题、认识问题，并根据这些事物的本质特点，把具有相似内部状态和运动规律的实体（事物、对象）综合在一起称为类，类是具有相似内部状态和运动规律的实体的抽象，按人们认识客观世界的系统思维方式，采用基于对象（实体）的概念建立模型，模拟客观世界分析、设计、实现软件的办法。

15.1.1　面向对象的基本概念

1. 对象（Object）

对象是人们要进行研究的任何事物，是指现实世界中各种各样的实体。每个对象皆有自己的

内部状态和行为规律：

- 对象具有状态，可用数据值来描述它的状态。
- 对象具有操作，用于改变对象的状态，对象及其操作就是对象的行为。
- 对象实现了数据和操作的结合，使数据和操作封装于对象的统一体中。

2. 类（Class）

类是具有相似内部状态和运动规律的实体的集合。类的概念来自于人们认识自然、认识社会的过程，因此，对象的抽象是类，类的具体化就是对象。

- 类具有属性，它是对象的状态的抽象，用数据结构来描述类的属性。
- 类具有操作，它是对象的行为的抽象，用操作名和实现该操作的方法来描述。
- 类具有结构，类通常有两种主要的结构关系，即一般和整体结构关系。① 一般：具体结构称为分类结构，也可以说是“或”关系，或者是“is a”关系。② 整体：部分结构称为组装结构，它们之间的关系是一种“与”关系，或者是“has a”关系。

3. 消息（Message）

消息是指对象之间进行通信的结构叫做消息。当一个消息发送给某个对象时，消息包含 5 部分：

- 发送消息的对象。
- 接收消息的对象。
- 消息传递的办法。
- 消息内容（该消息的对象所知道的变量名）。
- 反馈。

15.1.2 类的特性

类具有以下 6 个特性：惟一性、分层性、继承性、封装性、多态性、重载性、共享性。

1. 惟一性

每个对象都有自身惟一的标识，通过这种标识，可找到相应的对象。在对象的整个生命期中，它的标识都不改变，不同的对象不能有相同的标识。

2. 分层性

类可分为以下三个层次。

- 概念层：概念层是一个较高的层次，通常在进行领域分析时，为了建立概念模型时使用。在这个层次中，类只是一个概念。
- 规格层：在规格层，类已经是属于软件开发范畴了，但主要关注的是类的接口，而不是类的实现。
- 实现层：在实现层关注类的实现，如利用某种语言写成的函数体、定义的成员变量等。

3. 继承性

继承性是面向对象程序设计语言不同于其他语言的最重要的特点，是其他语言所没有的。继承描述的是一种抽象到具体的关系。继承不但利用了抽象的力量来降低系统的复杂性，它还提供了一种重用的方式。类可以分为父类和子类，继承可以分为单重继承和多重继承。

- 子类只继承一个父类的数据结构和方法，则称为单重继承。
- 子类继承了多个父类的数据结构和方法，则称为多重继承。

类的继承性使所建立的软件具有开放性、可扩充性，这是信息组织与分类的行之有效的方法，它简化了对象、类的创建工作量，增加了代码的可重用性。

4. 封装性

对象间的相互联系和相互作用过程主要通过消息机制得以实现。对象之间不需要了解对方内部的具体状态或运动规律。面向对象的类是封装良好的模块，类定义将其说明（用户可见的外部接口）与实现（用户不可见的内部实现）显式地分开，其内部实现按其具体定义的作用域提供保护。类是封装的最基本单位。封装防止了程序相互依赖性而带来的变动影响。在类中定义，接收对方消息的方法称为类的接口。

5. 多态性（覆盖）

多态性是指同名的方法可在不同的类中具有不同的运动规律。多态性是指相同的操作或函数、过程可作用于多种类型的对象上并获得不同的结果。不同的对象，收到同一消息可以产生不同的结果，这种现象称为多态性。

在父类演绎为子类时，类的运动规律也同样可以演绎，演绎使子类的同名运动规律或运动形式更具体，甚至子类可以有不同于父类的运动规律或运动形式。不同的子类可以演绎出不同的运动规律。

6. 重载性

重载是指类的同名方法在传递不同的参数时可以有不同的运动规律。在对象间相互作用时，即使接收消息对象采用相同的接收办法，但消息内容的详细程度不同，接收消息对象内部的运动规律也可能不同。

7. 共享性

面向对象技术在不同级别上促进了共享。

- 同一类中的共享：同一类中的对象有着相同的数据结构，这些对象之间是结构、行为特征的共享关系。
- 在同一应用中共享：在同一应用的类层次结构内存在继承关系的各相似子类中，存在数据结构和行为的继承，使各相似子类共享共同的结构和行为。使用继承来实现代码的共享。
- 在不同应用中共享：面向对象不仅允许在同一应用中共享信息，而且为未来目标的可重用设计准备了条件。通过类库这种机制和结构来实现不同应用中的信息共享。

15.1.3 面向对象的开发方法

面向对象技术出现了几十种支持软件开发的面向对象方法，其中 Booch、Coad、OMT、UML 和 Jacobson 的方法在面向对象软件开发界得到了广泛认可。

1. Booch 方法

Booch 最先描述了面向对象的软件开发方法的基础问题，指出面向对象开发是一种根本不同于传统的功能分解的设计方法。面向对象的软件分解更接近于人对客观事务的理解，而功能分解只通过问题空间的转换来获得。

Booch 方法的过程包括以下步骤：

- 在给定的抽象层次上识别类和对象。
- 识别这些对象和类的语义。
- 识别这些类和对象之间的关系。
- 实现类和对象。

这 4 种活动不仅仅是一个简单的步骤序列，而是对系统的逻辑和物理视图不断细化的迭代和渐增的开发过程。

Booch 方法的符号体系如下：

- 类图（类结构－静态视图）。
- 对象图（对象结构－静态视图）。
- 状态转移图（类结构－动态视图）。
- 时态图（对象结构－动态视图）。
- 模块图（模块体系结构）。
- 进程图（进程体系结构）。

用于类和对象建模的符号体系使用注释和不同的图符（如不同的箭头）表达详细的信息。Booch 建议在设计的初期可以用符号体系的一个子集，随后不断添加细节。对每一个符号体系还有一个文本的形式，由每一个主要结构的描述模板组成。符号体系由大量的图符定义，但是，其语法和语义并没有严格地定义。

2. Coad 方法

Coad 方法是 1989 年由 Coad 和 Yourdon 提出的面向对象开发方法。该方法的主要优点是通过大系统开发的经验与面向对象概念的有机结合，在对象、结构、属性和操作的认定方面，提出了一套系统的原则。

Coad 方法完成了从需求角度进一步进行类和类层次结构的认定。Coad 方法将开发分为面向对象分析（OOA）、面向对象设计（OOD）和面向对象编程（OOP）三个阶段。

（1）面向对象分析（OOA）阶段

面向对象分析（OOA）阶段的活动主要包括发现类及对象、识别结构、定义主题、定义属性、定义服务等。

- 发现类及对象：发现类及对象是描述如何发现类及对象。从应用领域开始识别类及对象，形成整个应用的基础，然后据此分析系统的责任。
- 识别结构：识别结构分为两个步骤：① 识别一般-特殊结构，该结构捕获了识别出的类的层次结构；② 识别整体-部分结构，该结构用来表示一个对象如何成为另一个对象的一部分，以及多个对象如何组装成更大的对象。
- 定义主题：主题由一组类及对象组成，用于将类及对象模型划分为更大的单位，便于理解。
- 定义属性：定义属性包括定义类的实例（对象）之间的实例连接。
- 定义服务：定义服务包括定义对象之间的消息连接。在面向对象分析阶段产生整个问题空间的抽象描述，在此基础上，进一步归纳出适用于面向对象编程语言的类和类结构，最后形成代码。由于面向对象的特点，采用这种开发模型能有效地将分析设计的文本或图表代码化。

（2）面向对象设计（OOD）阶段

面向对象设计（OOD）采用“造型的观点”，以 OOA 为基础归纳出类，并建立类结构或进一步构造成类库，实现分析结果对问题空间的抽象。OOD 确定类和类结构不仅是满足当前需求分析的要求，更重要的是通过重新组合或加以适当的补充，能方便实现功能的重用和扩增，以不断适应用户的要求。

面向对象设计模型主要分为 4 个部分。

- 问题域部分（PDC）：面向对象分析的结果直接放入该部分。
- 人机交互部分（HIC）：这部分的活动包括对用户分类、描述人机交互的脚本、设计命令层次结构、设计详细的交互、生成用户界面的原型、定义 HIC 类。
- 任务管理部分（TMC）：这部分的活动包括识别任务（进程）、任务所提供的服务、任务的优先级、进程是事件驱动还是时钟驱动以及任务与其他进程和外界如何通信。
- 数据管理部分（DMC）：这一部分依赖于存储技术，是文件系统、关系数据库管理系统还是面向对象数据库管理系统。

（3）面向对象编程（OOP）阶段

面向对象程序具有继承、封装和多态的新特性。继承是面向对象程序的重要特点，继承使得代码的重用率提高，同时也使错误传播的概率提高。

面向对象程序是把功能的实现分布在类中。能正确实现功能的类，通过消息传递来协同实现设计要求的功能。正是这种面向对象程序风格，将出现的错误能精确地确定在某一具体的类，因此，在面向对象编程（OOP）阶段，忽略类功能实现的细则，主要体现为以下两个方面。

- 数据成员是否满足数据封装：数据封装是数据和数据有关的操作的集合。检查数据成员是否满足数据封装的要求，基本原则是数据成员是否被外界直接调用。当改变数据成员的结构时，是否影响了类的对外接口，是否会导致相应外界必须改动。有时强制的类型转换会破坏数据的封装特性。
- 类是否实现了要求的功能：类所实现的功能，都是通过类的成员函数执行的。在功能实现时，应该首先保证类成员函数的正确性；单独的看待类的成员函数与面向过程程序中的函数或过程没有本质的区别。类函数成员的正确行为只是类能够实现要求的功能的基础。

3. OMT 方法

OMT 方法是 1991 年由 James Rumbaugh 等 5 人提出的，该方法是一种新兴的面向对象的开发方法，开发工作的基础是对真实世界的对象建模，然后围绕这些对象使用分析模型来进行独立于语言的设计，面向对象的建模和设计促进了对需求的理解，有利于开发出更清晰、更容易维护的软件系统。该方法为大多数应用领域的软件开发提供了一种实际的、高效的保证，努力寻求一种问题求解的实际方法。

OMT 方法提供了三种模型，即对象模型、动态模型和功能模型。

- 对象模型主要描述对象的静态结构和它们之间的关系，对象模型的主要概念包括类、属性、操作、继承、关联（即关系）、聚集等。
- 动态模型主要描述系统中随时间变化的方面，动态模型的主要概念包括状态、子状态、超状态、事件、行为、活动。
- 功能模型主要描述系统内部数据值的转换，功能模型的主要概念有加工、数据存储、数据流、控制流、角色。

OMT 方法将开发过程分为 4 个阶段，即分析阶段、系统设计阶段、对象设计阶段、实现阶段。

- 分析阶段：基于问题和用户需求的描述，建立现实世界的模型。分析阶段的主要产物有问题描述；对象模型=对象图+数据词典；动态模型=状态图+全局事件流图；功能模型=数据流图+约束。
- 系统设计阶段：结合问题域的知识和目标系统的体系结构（求解域），将目标系统分解为子系统。
- 对象设计阶段：基于分析模型和求解域中的体系结构等添加的实现细节，完成系统设计。主要产物包括：细化的对象模型、细化的动态模型、细化的功能模型。
- 实现阶段：将设计转换为特定的编程语言或硬件，同时保持可追踪性、灵活性和可扩展性。

4. UML 建模语言

UML（Unified Modeling Language）建模语言将是面向对象技术领域内占主导地位的标准建模语言。UML 结合了 Booch 方法、OMT 方法、和 Jacobson 方法的优点，统一了符号体系，并从其他的方法和工程实践中吸收了许多经过实际检验的概念和技术，而且对其作了进一步的发展，最终统一为大众接受的标准建模语言。UML 是一种定义良好、易于表达、功能强大且普遍适用的建模语言。它融入了软件工程领域的新思想、新方法和新技术。它的作用域不限于支持面向对象的分析与设计，还支持从需求分析开始的软件开发全过程。

5. Jacobson 方法

Jacobson 方法与上述 4 种方法有所不同，它涉及到整个软件生命周期，包括需求分析、设计、实现和测试等 4 个阶段。

需求分析阶段的活动包括定义潜在的角色（角色是指使用系统的人和与系统互相作用的软、硬件环境），识别问题域中的对象和关系，基于需求规范说明和角色的需要发现 use case，并详细描述 use case。

设计阶段主要包括两个活动，从需求分析模型中发现设计对象，以及针对实现环境调整设计

模型。第一个活动包括从 use case 的描述发现设计对象，并描述对象的属性、行为和关联。在这里还要把 use case 的行为分派给对象。

在需求分析阶段的识别领域对象和关系的活动中，开发人员识别类、属性和关系。关系包括继承、组成（聚集）和通信关联。定义 use case 的活动和识别设计对象的活动，两个活动共同完成行为的描述。Jacobson 方法还将对象区分为语义对象（领域对象）、界面对象（如用户界面对象）和控制对象（处理界面对象和领域对象之间的控制）。

在该方法中的一个关键概念就是 use case。use case 是指与行为相关的事务序列，该序列将由用户在与系统对话中执行，因此，每一个 use case 就是一个使用系统的方式，当用户给定一个输入，就执行一个 use case 的实例并引发执行属于该 use case 的一个事务。

15.1.4　面向对象的模型

面向对象的模型主要可分为对象模型、动态模型、功能模型。

1. 对象模型

对象模型用于表示静态的、结构化的系统数据性质，描述了系统的静态结构，它是从客观世界实体的对象关系角度来描述，表现了对象的相互关系。对象模型主要关心系统中对象的结构、属性和操作。

对象模型是由一个或若干个模板组成。模板将模型分为若干个便于管理的子块，在整个对象模型、类及关联的构造块之间，模板提供了一种集成的中间单元，模板中的类名及关联名是惟一的。

（1）对象和类

对象建模的目的就是描述对象。通过将对象抽象成类，使问题抽象化，从而增强模型的归纳能力。属性指的是类中对象所具有的性质（数据值）。

操作是类中对象所使用的一种功能或变换。类中的各对象可以共享操作，每个操作都有一个目标对象作为其隐含参数。方法是类的操作的实现步骤。

（2）关联和链

关联是建立类之间关系的一种手段，而链则是建立对象之间关系的一种手段。

- 关联和链的含义：链表示对象间的物理与概念联结，关联表示类之间的一种关系，链是关联的实例，关联是链的抽象。
- 角色：角色说明类在关联中的作用，它位于关联的端点。
- 受限关联：受限关联由两个类及一个限定词组成，限定词是一种特定的属性，用来有效地减少关联的重数，限定词在关联的终端对象集中说明。限定提高了语义的精确性，增强了查询能力，在现实世界中，常常出现限定词。
- 关联的多重性：关联的多重性是指类中有多少个对象与关联的类的一个对象相关。

（3）类的层次结构

聚集是一种“整体-部分”关系。在这种关系中，有整体类和部分类之分。聚集最重要的性质是传递性，也具有逆对称性。聚集可以有不同的层次，可以把不同分类聚集起来。

一般化关系是在保留对象差异的同时共享对象相似性的一种高度抽象方式。

（4）对象模型

模板是类、关联、一般化结构的逻辑组成。对象模型是由一个或若干个模板组成。模板将模型分为若干个便于管理的子块，在整个对象模型、类及关联的构造块之间，模板提供了一种集成的中间单元，模板中的类名及关联名是惟一的。

2. 动态模型

动态模型是与时间和变化有关的系统性质。该模型描述了系统的控制结构，表示了瞬间的、行为化的系统控制性质，它关心的是系统的控制、操作的执行顺序，表示从对象的事件和状态的角度出发，表现了对象的相互行为。

动态模型描述的系统属性是触发事件、事件序列、状态、事件与状态的组织。使用状态图作为描述工具，涉及到事件、状态、操作等重要概念。

（1）事件

事件是指定时刻发生的某件事。

（2）状态

状态是对象属性值的抽象。对象的属性值按照影响对象显著行为的性质将其归并到一个状态中去。状态指明了对象对输入事件的响应。

（3）状态图

状态图反映了状态与事件的关系。当接收一事件时，下一状态就取决于当前状态和所接收的该事件，由该事件引起的状态变化称为转换。

状态图是一种图，用结点表示状态，结点用圆圈表示；圆圈内有状态名，用箭头连线表示状态的转换，上面标记事件名，箭头方向表示转换的方向。

3. 功能模型

功能模型描述了系统的所有计算。功能模型指出发生了什么，动态模型确定什么时候发生，而对象模型确定发生的客体。功能模型表明一个计算如何从输入值得到输出值，它不考虑计算的次序。功能模型由多张数据流图组成。数据流图用来表示从源对象到目标对象的数据值的流向，它不包含控制信息，控制信息在动态模型中表示，同时数据流图也不表示对象中值的组织，值的组织在对象模型中表示。

数据流图中包含有处理、数据流、动作对象和数据存储对象。

（1）处理

数据流图中的处理用来改变数据值。最低层处理是纯粹的函数，一张完整的数据流图是一个高层处理。

（2）数据流

数据流图中的数据流将对象的输出与处理、处理与对象的输入、处理与处理联系起来。用数

据流来表示一中间数据值，数据流不能改变数据值。

（3）动作对象

动作对象是一种主动对象，它通过生成或者使用数据值来驱动数据流图。

（4）数据存储对象

数据流图中的数据存储是被动对象，它用来存储数据。它与动作对象不一样，数据存储本身不产生任何操作，它只响应存储和访问的要求。

15.1.5 面向对象的设计

面向对象设计是把分析阶段得到的需求转变成符合成本和质量要求的、抽象的系统实现方案的过程。设计应注意以下几点。

1. 面向对象设计的准则

（1）模块化

面向对象开发方法把系统分解成模块。

（2）抽象

面向对象方法不仅支持过程抽象，而且支持数据抽象。

（3）信息隐藏

在面向对象方法中，信息隐藏通过对象的封装性来实现。

（4）低耦合

耦合主要指不同对象之间相互关联的紧密程度，低耦合是设计的一个重要标准，因为这有助于使得系统中某一部分的变化对其他部分的影响降到最低程度。

（5）高内聚

高内聚具有以下三重含义：

- 操作内聚。
- 类内聚。
- 一般具体结构的内聚。

2. 面向对象设计的规则

（1）设计结果

设计结果应该清晰易懂，需要做到如下几点：

- 用词一致。
- 使用已有的协议。
- 减少消息模式的数量。

- 避免模糊的定义。

（2）设计简单类

设计简单类时应该注意以下几点：

- 避免包含过多的属性。
- 有明确的定义。
- 尽量简化对象之间的合作关系。
- 不要提供太多的操作。

（3）使用简单的协议

一般来说，消息中参数不要超过 3 个。

（4）使用简单的操作

面向对象设计出来的类中的操作通常都很小，一般只有 3～5 行源程序语句，可以用仅含一个动词和一个宾语的简单句子描述它的功能。

（5）把设计变动减至最小

通常，设计的质量越高，设计结果保持不变的时间也越长，即使出现必须修改设计的情况，也应该使修改的范围尽可能小。

15.1.6　面向对象的测试内容

面向对象的测试与传统测试的目标是一致的，面向对象把测试扩大到分析和设计模型的评审，面向对象测试的不是模块，而是类。

1. 传统测试模式与面向对象测试模式的主要区别

传统测试模式与面向对象测试模式的最主要区别在于：面向对象的测试更关注对象，而不是完成输入/输出的单一功能，这样的话测试可以在分析与设计阶段就先行介入，使得测试更好地配合软件生产过程并为之服务。

2. 面向对象测试的主要内容

面向对象测试的主要内容如下：

- 面向对象分析的测试（OOA Test）。
- 面向对象设计的测试（OOD Test）。
- 面向对象编程的测试（OOP Test）。
- 面向对象单元测试（OO Unit Test）。
- 面向对象集成测试（OO Integration Test）。
- 面向对象系统测试（OO System Test）。

面向对象测试的另一种划分：模型测试、类测试（用于代替单元测试）、交互测试（用于代替集成测试）、系统（包括子系统）测试、接收测试、部署测试。

3. 面向对象的测试过程

面向对象的测试过程如下：

- 指定范围。
- 指定深度。
- 指定已创建的被测试模块的基本要求（上一个阶段需要提供的接口）。
- 以基本模型的内容为输入来设计测试用例作为评估标准。
- 生成测试覆盖度量标准。
- 试用测试清单执行静态分析，确保被测模块与基本模型的一致性。
- 执行测试用例。
- 如果覆盖不足以检测所有的活动，就需要分解测试工作，并且使用传统测试用例的方式来警醒，或者中断测试，重新测试传统测试用例。

4. 面向对象的测试方法

面向对象的测试方法主要有如下 5 种。

（1）基于故障的测试

基于故障的测试包括如下内容：

- 发现可能故障的能力。
- 从分析模型开始，考察可能发生的故障，设计用例去执行设计和代码。
- 从客户对象（主动）上发现错误。
- 不能发现的错误包括：不正确的规格说明、用户不需要的功能或缺少用户需要的功能、没有考虑子系统间的交互作用。

（2）基于场景的测试

基于场景的测试主要关注用户需要做什么，从用户任务（使用用例）中找出用户要做什么及如何去执行。

在一个单元测试情况下有助于帮助检查多重系统。

（3）OO 类的随机测试

如果一个类有多个操作（功能），这些操作（功能）序列有多种排列，这种不变化的操作序列可随机产生，利用这种可随机排列来检查不同类实例的生存史，称为随机测试。

（4）类层次的分割测试

类层次的分割测试可以减少用完全相同的方式检查类测试用例的数目；分类可分为基于状态的分割、基于属性的分割、基于类型的分割。

- 基于状态的分割：按类操作是否会改变类的状态进行分割（归类）。
- 基于属性的分割：按类操作所得到的属性来分割（归类）。
- 基于类型的分割：按完成的功能分割（分类），如初始操作、计算操作、查询操作。

（5）由行为模型（状态、活动、顺序）导出的测试

行为模型可以用来帮助导出类的动态行为的测试序列，以及这些类与之合作的类的动态行为测试用例，设计出最小测试用例，保证类的所有行为被充分检查。

5. 面向对象的测试优点

面向对象测试的优点如下：

- 更早地定义出测试用例。
- 早期介入可以降低成本。
- 尽早地编写系统测试用例，以便于开发人员与测试人员对系统需求的理解保持一致。
- 面向对象的测试模式更注重于软件的实质。

15.1.7　面向对象的测试模型

面向对象将开发分为面向对象分析（OOA）、面向对象设计（OOD）和面向对象编程（OOP）三个阶段。分析阶段产生整个问题空间的抽象描述，在此基础上，进一步归纳出适用于面向对象编程语言的类和类结构，最后形成代码。由于这种开发特点，面向对象测试结合传统的测试步骤的划分，测试模型如图 15-1 所示。

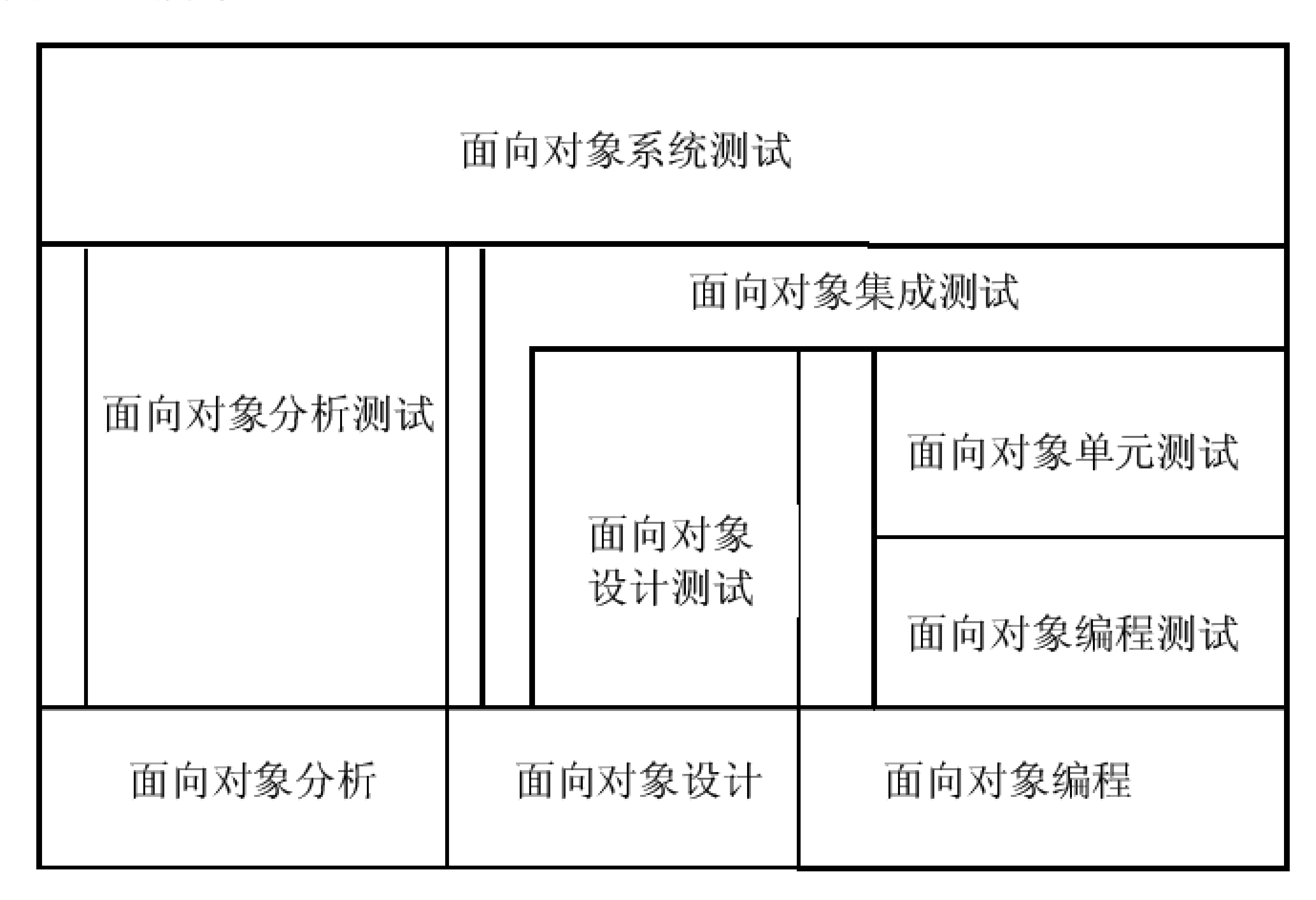


图 15-1　面向对象测试模型

面向对象分析测试和面向对象设计测试是对分析结果和设计结果的测试，主要是对分析设计产生的文本进行的，是软件开发前期的关键性测试。

重点如下：

- 表示分析模型和设计模型的符号体系和语法检查。
- 专家评审类定义符合现实问题的表示程度，有无遗漏和歧义性的检查。
- 面向对象分析和面向对象设计模型的表示一致性。

面向对象编程测试主要针对编程风格和程序代码的实现进行测试，其主要的测试内容在面向对象单元测试和面向对象集成测试中体现。

面向对象单元测试是对程序内部具体单一模块的测试，主要就是对类成员函数的测试。面向对象单元测试是进行面向对象集成测试的基础。

面向对象集成测试主要对系统内部的相互服务进行测试，如成员函数间的相互作用、类间的消息传递等。面向对象集成测试不但要基于面向对象的单元测试，更要参考面向对象设计测试结果。

面向对象系统测试是基于面向对象集成测试的最后阶段的测试，主要以用户需求为测试标准，需要借鉴面向对象分析测试结果。

1. 整体流程

面向对象的测试整体流程如图 15-2 所示。

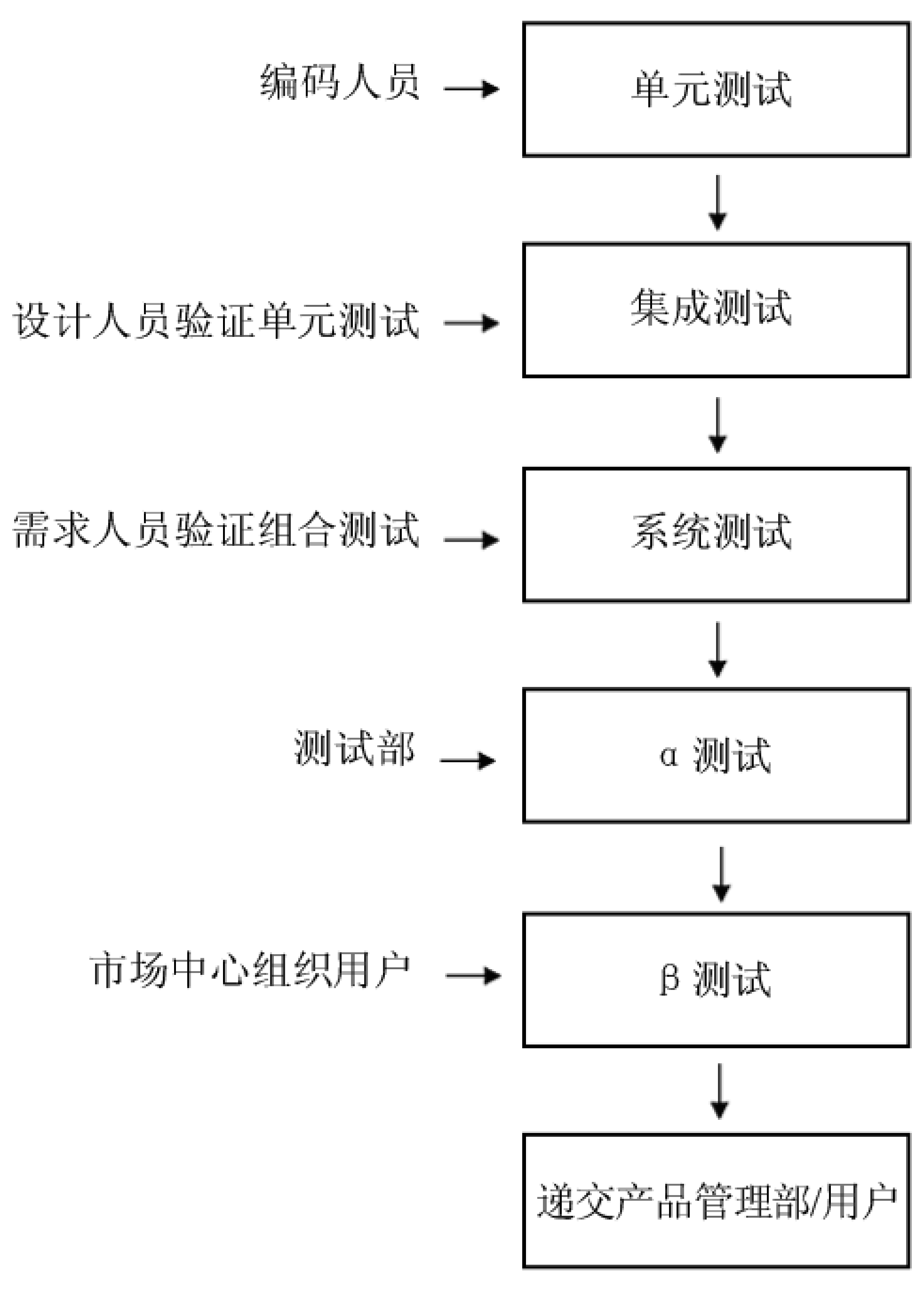


图 15-2　面向对象的测试整体流程图

对图 15-2 的说明如下：

- 编码人员进行单元测试工作，以规范的格式提交可运行的程序模块。
- 单元测试结果由设计人员进行核实和验证。
- 设计人员确认移交产品各项功能完备和稳定性之后，进行集成测试。
- 集成测试结果提交需求人员进行核实和验证。
- 需求人员确认移交产品各项功能完备和稳定性之后，进行系统测试。
- 系统测试结果经产品/项目经理核实和验证后，提交测试部进行 α 测试。
- 市场中心组织用户进行 β 测试。

2. 测试重点

测试的重点是：测试工作的工期、人员安排、设备和测试环境的建立、测试记录统计、问题反馈、纠错处理、接口状态描述、测试用例及其数据描述、测试计划、需求说明书、OOA 文档、OOD 文档、测试记录、用户反馈意见记录等。

3. 测试记录

测试记录时具有如下要求：

- 问题描述正确。
- 错误过程记录完整。
- 界面提示记录完整易懂。
- 错误出现操作步骤详细。

4. 测试记录表

测试记录表如表 15-1 所示。

表 15-1　测试记录表

编号

产品/模块名称：		版本号：		测试时间：	
错误种类：		功能号：		发现日期：	
现象： 备注说明：					
开发人员反馈： 解决日期：　　　　解决人：					
备注说明： 测试人员：　　　　负责人：					

15.2　面向对象分析的测试

面向对象分析是把 E-R 图和语义网络模型，即信息造型中的概念和面向对象程序设计语言中的重要概念结合在一起而形成的分析方法，最后通常是得到问题空间的图表的形式描述。

分析是直接映射问题空间，全面地将问题空间中实现功能的现实抽象化。将问题空间中的实例抽象为对象，用对象的结构反映问题空间的复杂实例和复杂关系，用属性和服务表示实例的特性和行为。

1. 面向对象分析测试的 5 个方面

面向对象分析的测试（OOA Test）可以划分为以下 5 个方面：

- 对认定的对象的测试。
- 对认定的结构的测试。
- 对认定的主题层的测试。
- 对定义的属性和实例关联的测试。
- 对定义的服务和消息关联的测试。

测试的重点在于其完整性和冗余性。

（1）对认定的对象的测试

面向对象分析认定的对象是对问题空间中的结构、其他系统、设备、被记忆的事件、系统涉及的人员等实际实例的抽象。测试应重点注意如下内容：

- 认定的对象是否全面，是否问题空间中任何涉及到的实例都反映在认定的抽象对象中。
- 认定的对象是否具备多个属性。只有一个属性的对象通常应看成是其他对象的属性，而不是抽象为单独的对象。
- 对认定为同一对象的实例是否提供或需要相同的服务，假如服务随着不同的实例而变化，认定的对象就需要分解或利用继承性来分类表示。
- 假如系统没有必要始终保持对象代表的实例信息，提供或得到关于它的服务，认定的对象也无必要。
- 认定的对象名称应该尽量准确、适用。

（2）对认定的结构的测试

认定的结构指的是多种对象的组织方式，用来反映问题空间中的复杂实例和复杂关系。认定的结构可分为两种：分类结构和组装结构。

- 分类结构体现了问题空间中实例的一般和特别的关系。
- 组装结构体现了问题空间中实例整体和局部的关系。

① 对认定的分类结构的测试

对认定的分类结构的测试时应该注意如下内容：

- 对于结构中的一种对象，尤其是处于高层的对象，是否在问题空间中含有不同于下一层对象的特别可能性，即是否能派生出下一层对象。
- 对于结构中的一种对象，尤其是处于同一低层的对象，是否能抽象出在现实中有意义的、更一般的上层对象。
- 对任何认定的对象，是否能在问题空间内向上层抽象出在现实中有意义的对象。
- 高层的对象的特性是否完全体现下层的共性。
- 低层的对象是否有高层特性基础上的特性。

② 对认定的组装结构的测试

对认定的组装结构的测试时应注意如下内容：

- 整体（对象）和部件（对象）的组装关系是否符合现实的关系。
- 整体（对象）的部件（对象）是否在考虑的问题空间中有实际应用。
- 整体（对象）中是否遗漏了反映在问题空间中有用的部件（对象）。
- 部件（对象）是否能够在问题空间中组装新的有现实意义的整体（对象）。

（3）对认定的主题层的测试

主题层是在对象和结构的基础上更高一层的抽象，是为了提供面向对象分析结果的可见性，如同文章对各部分内容的概要。对主题层的测试应注意如下内容：

- 贯彻 George Miller 的“7+2”原则，假如主题个数超过 7 个，就需要对有较密切属性和服务的主题进行归并。
- 主题所反映的一组对象和结构是否具备相同、相近的属性和服务。
- 认定的主题是否是对对象和结构更高层地抽象，是否便于理解 OOA 结果的概貌。
- 主题间的消息联系（抽象）是否代表了主题所反映的对象和结构之间的任何关联。

（4）对定义的属性和实例关联的测试

属性是用来描述对象或结构所反映的实例的特性，而实例关联是反映实例集合间的映射关系。对属性和实例关联的测试应注意如下内容：

- 定义的属性是否对相应的对象和分类结构的每个现实实例都适用。
- 定义的属性在现实世界中是否和这种实例关系密切。
- 定义的属性在问题空间是否和这种实例关系密切。
- 定义的属性是否能够不依赖于其他属性被单独理解。
- 定义的属性在分类结构中的位置是否恰当，低层对象的共有属性是否在上层对象属性中体现。
- 在问题空间中每个对象的属性是否定义完整。
- 定义的实例关联是否符合现实。
- 在问题空间中实例关联是否定义完整，特别需要注意 1-多和多-多的实例关联。

（5）对定义的服务和消息关联的测试

定义的服务就是定义的每一种对象和结构在问题空间所需要的行为。由于问题空间中实现必要的通信，在面向对象分析中相应需要定义消息关联。对定义的服务和消息关联的测试应注意如下内容：

- 对象和结构在问题空间的不同状态是否定义了相应的服务。
- 对象或结构所需要的服务是否都定义了相应的消息关联。
- 定义的消息关联所指引的服务提供是否正确。
- 沿着消息关联执行的线程是否合理，是否符合现实过程。
- 定义的服务是否重复，是否定义了能够得到的服务。

2. 面向对象分析的缺点

面向对象分析的缺点如下：

- 对问题空间分析抽象的不完整，会影响软件的功能实现，导致软件开发后期产生大量原本可避免的修补工作。
- 一些冗余的对象或结构类的选定，将增加程序员不必要的工作量。

3. 面向对象设计的测试

面向对象设计（OOD）采用“造型的观点”，以 OOA 为基础归纳出类，并建立类结构或进一步构造成类库，实现分析结果对问题空间的抽象。OOD 归纳的类既可以是对象的简单延续，也可以是不同对象的相同或相似的服务。

面向对象设计的测试（OOD Test）可以划分为以下三个方面：

- 对认定的类的测试。
- 对构造的类层次结构的测试。
- 对类库的支持的测试。

（1）对认定的类的测试

OOD 认定的类可以是 OOA 中认定的对象，也可以是对象所需要的服务的抽象，所以 OOD、OOA 认定的类的界限很难区分。OOD 确定类和类结构不仅是满足当前需求分析要求，更重要的是通过重新组合或加以适当地补充，方便实现功能重用和扩增，因此，对 OOD 的测试，建议针对功能的实现和重用以及 OOA 结果的分析。

认定的类原则上应该尽量基础性，这样才能便于维护和重用。对认定的类的测试应重点注意如下 6 点内容：

- 是否涵盖了 OOA 中任何认定的对象。
- 是否能体现 OOA 中定义的属性。
- 是否能实现 OOA 中定义的服务。
- 是否对应着一个含义明确的数据抽象。
- 是否尽可能少的依赖其他类。
- 类中的方法（C++: 类的成员函数）是否单用途。

（2）对构造的类层次结构的测试

为能充分发挥面向对象的继承共享特性，OOD 的类层次结构，通常基于 OOA 中产生的分类结构的原则来组织，着重体现父类和子类间一般性和特别性。在当前的问题空间，对类层次结构的主要需要是能在解空间构造实现全部功能的结构框架。为此，测试应重点注意如下 6 点内容：

- 类层次结构是否涵盖了任何定义的类。
- 是否能体现 OOA 中所定义的实例关联。
- 是否能实现 OOA 中所定义的消息关联。
- 子类是否具备父类没有的新特性。
- 子类间的共同特性是否完全在父类中得以体现。

（3）对类库的支持的测试

对类库的支持虽然也属于类层次结构的组织问题，但其强调的重点是再次软件研发的重用。由于它并不直接影响当前软件的研发和功能实现，因此，可以将其单独提出来测试，也可作为对高质量类层次结构的评估。测试时应重点注意如下要点：

- 一组子类中关于某种含义相同或基本相同的操作，是否有相同的接口（包括名字和参数表）。
- 类中方法功能是否较单纯，相应的代码行是否较少。
- 类的层次结构是否是深度大、宽度小。

15.3　面向对象编程的测试

面向对象程序具有继承、封装和多态的新特性，这使得传统的测试策略必须有所改变。重点内容如下。

- 封装：封装是对数据的隐藏，外界只能通过被提供的操作来访问或修改数据，这样降低了数据被任意修改和读写的可能性，降低了传统程序中对数据非法操作的测试。
- 继承：继承是面向对象程序的重要特点，继承使得代码的重用率提高，同时也使错误传播的概率提高。
- 多态：多态使得面向对象程序对外呈现出强大的处理能力，但同时却使得程序内“同一”函数的行为复杂化，测试时不得不考虑不同类型具体执行的代码和产生的行为。

面向对象程序是把功能的实现分布在类中。能正确实现功能的类，通过消息传递来协同实现设计要求的功能，因此，在面向对象编程（OOP）阶段，忽略类功能实现的细则，将测试的目光集中在类功能的实现和相应的面向对象程序风格上，主要体现为以下两个方面。

- 数据成员是否满足数据封装的要求。
- 类是否实现了要求的功能。

1. 数据成员是否满足数据封装的要求

数据封装是数据和数据有关的操作的集合。检查数据成员是否满足数据封装的要求，基本原则是数据成员是否被外界（数据成员所属的类或子类以外的调用）直接调用。更直观地说，当改变数据成员的结构时，是否影响了类的对外接口、是否会导致相应外界必须改动。值得注意的是，有时强制的类型转换会破坏数据的封装特性。

2. 类是否实现了要求的功能

类所实现的功能都是通过类的成员函数执行的。在测试类的功能实现时，应该首先保证类成员函数的正确性。单独的看待类的成员函数，与面向过程程序中的函数或过程没有本质的区别，几乎所有传统的单元测试中所使用的方法，都可在面向对象的单元测试中使用。具体的测试方法在面向对象的单元测试中介绍。类函数成员的正确行为只是类能够实现要求功能的基础，类成员函数间的作用和类之间的服务调用是单元测试无法确定的，因此，需要进行面向对象的集成测试。具体的测试方法在面向对象的集成测试中介绍。需要着重声明的是，测试类的功能，不能仅满足于代码能

无错运行或被测试类能提供的功能无错，应该以所做的 OOD 结果为依据，检测类提供的功能是否满足设计的要求、是否有缺陷。必要时还应该参照 OOA 的结果，以之为最终标准。

15.4 面向对象的单元测试

15.4.1 类的测试和测试要求

1. 类的测试

传统的单元测试的对象是软件设计的最小单位——模块。单元测试的依据是详细设计描述，单元测试应对模块内所有重要的控制路径设计测试用例，以便发现模块内部的错误。面向对象的单元测试（OO Unit Test）是对 OO 软件的类测试（等价于传统软件的单元测试），它关注模块的算法细节和模块接口间流动的数据，它是由封装在类中的操作和类的状态行为驱动的。

面向对象软件的最小单位是类，这意味着每个类和类的实例（对象）封装了属性（数据）和操纵这些数据的操作。最小的可测试单位是封装的类或对象，类包含一组不同的操作，并且某特殊操作可能作为一组不同类的一部分存在，因此，单元测试的意义发生了较大变化。我们不再孤立地测试单个操作，而是将操作作为类的一部分，一般建议由程序员完成。

2. 测试要求

类的测试要求如下：

- 若类的测试不足，则很多错误不能被查出。
- 通过客户类去测试服务类是很困难的。
- 错误发现越晚，纠正的代价越大。

有效的类测试可减少开发时间，并提提高生产率。

15.4.2 类测试设计的方法

类测试设计的方法可以分为方法范围的测试设计、类范围的测试设计、继承的测试设计。

1. 方法范围的测试设计

（1）方法范围测试设计的基本过程

方法范围测试设计的基本过程如下：

01 设置测试消息的参数、类变量、全局变量为希望的状态。
02 设置被测对象为希望的状态。
03 从驱动模块向该被测对象发送测试消息。
04 测试检查，测试检查时应注意如下内容：

- 比较被测对象返回的值和期望的值，若不同，则记录未通过。
- 比较被测对象的状态和期望的状态，若不同，则记录未通过。
- 若使用按引用调用，则比较测试消息参数的状态和期望的状态，若不同，则记录未通过。

- 捕获所有异常，假如是期望的异常，确定它是否正确。假如无异常或有一个不正确的异常，记录未通过。
- 假如一个断言的违反是期望的，确定它是否正确。若不是期望的，则记录未通过。
- 假如必要，比较类变量、全局变量的结果状态与期望的值，若有不期望的，则记录未通过。

05 假如发生任何未通过，则诊断和矫正该问题。

（2）方法范围测试设计的样式

方法范围测试设计的样式可以分为范畴划分、组合功能测试、递归功能测试、多态消息测试。对范畴划分的说明如下。

- 目标：根据输入/输出分析、设计方法范围测试包。
- 上下文和故障模型：假定故障与消息参数和实例变量值的组合有关，也假定这种故障将导致遗漏或错误的方法输出。
- 策略：标识每个功能的输入参数的范畴，范畴必须导致不同的输出；把每个范畴划分为选择；通过枚举所有选择的组合产生测试实例。

对组合功能测试的说明如下。

- 目标：为了根据消息值的组合选择状态，设计一个测试包。
- 上下文和故障模型：一个方法根据消息参数和实例变量的组合，选择很多不同动作中的一个时，选择输入组合以便适当地检测选择逻辑。
- 策略：对每个动作最少开发一个测试。

对递归功能测试的说明如下。

- 目标：对一个调用自己的方法，设计一个测试包。
- 上下文和故障模型：递归方法接收不正常的参数值、没有足够的栈空间或基本条件不能到达时，都会使递归失败。
- 策略：零次递归；一次递归；递归最大允许的或可行的深度。
- 破坏初始调用、下降阶段和上升阶段的前置、后置条件。

对多态消息测试的说明如下。

- 目标：多态服务执行所有客户与服务器的编联，对这种多态服务的一个客户开发测试。
- 上下文和故障模型：不能满足所有可能编联的前置条件；不希望的名字解析或指针的不正确构造，发生一个不希望的编联；服务类的实现发生了变化或扩充。
- 策略：对被测对象中每个多态消息的每种可能编联最少检测一次。

2. 类范围的测试设计

类范围的测试设计可分为不变量边界、非模态类测试、准模态类测试、模态类测试。

（1）不变量边界

对不变量边界的说明如下。

- 目标：为由复杂的和原始的数据类型组成的类、接口及构件选择能高效测试的测试值组合。

- 上下文和故障模型：为所有的变量提供值，随后运行一个测试；实例变量值的有效和无效的组合可被类的不变量指定。
- 策略：assert((txCounter>=0&&txCount<=5000)&&(creditlimit>99.99&&creditLimit<=1000000)&&((!account1.isClosed()||!account2.isClosed()))。

（2）非模态类测试

非模态类对接收的消息序列不强加任何限制，对非模态类测试的说明如下。

- 目标：为不限制消息顺序的类开发一个类范围测试包。
- 上下文和故障模型：拒绝一个合法的顺序；一个合法的顺序产生一个错误的值；接收非法的修改参数导致错误状态；错误的计算导致类的不变量被破坏等。
- 策略：随机序列；质疑序列。

（3）准模态类测试

准模态类对随对象的状态而改变的可接受消息进行顺序限制。对准模态类测试的说明如下。

- 目标：为类产生一个类范围测试包，该类对随类状态改变而改变的消息序列进行限制。
- 上下文和故障模型：容器和收集类通常是准模态的；有效的测试必须区别决定行为的内容和不影响行为的内容。
- 策略：定义不变量边界，集成操作序列。

（4）模态类测试

模态类对可接收的消息序列进行消息和域的限制。对模态类测试的说明如下。

- 目标：为一个在消息序列上有固定限制的类，开发一个类范围测试包。
- 上下文和故障模型：对可接收的消息序列和域两方面的限制；需要验证所有合法状态：将要被接受的消息被接受；在这些状态下非法的消息被拒绝；对接受或拒绝消息所得到的结果状态正确；对每个测试消息的响应是绝对正确的。
- 策略：基于状态的测试。

3. 继承的测试设计

继承的测试设计是为了确定继承的属性可在子类的环境中正确的运行。尽可能地复用父类测试包。对子类可以使用类范围内的测试。

- 设置测试消息的参数、类变量、全局变量为希望的状态。
- 设置被测对象为希望的状态。

15.4.3 单元测试使用的方法

单元测试使用的方法可分为故障测试、随机测试、划分测试。

1. 故障测试方法

故障测试方法是从分析模型开始查找可能的故障，对可能的故障设计案例，使之执行不正确的表达式，导致失败。

例如，正确写法：if(x>=0)calculate_the_square_root

错误写为：if(x>0)calculate_the_square_root

将会在边值处理时发生问题。

又如：正确写法：if(a || !b || c)

错误写为：if(a && !b || c)

将会在真假条件上发生错误。

这项技术的成功依赖于测试人员的感觉。

2. 随机测试方法

随机测试方法是用于分析类中最小的测试序列和最大的操作序列。在最小和最大序列之间，随机产生一系列不同的操作序列。

3. 划分测试方法

按状态、属性、操作、类级别分别归类所有的操作，设计测试案例，达到减少测试案例的功效。

- 状态划分测试方法：可定义的操作为状态操作、非状态操作。
- 属性划分测试方法：可定义的操作为使用 creditLimit、修改 creditLimit、不使用或不修改操作 credtLimit。
- 操作类别划分测试方法：可定义的操作为初始化操作、计算操作、查询操作、终止操作。
- 类级别划分测试方法：类级别的划分测试可以减少测试类所需要的测试案例数量、输入和输出被分类、设计测试案例来处理每个类别。

15.5 面向对象的集成测试

15.5.1 面向对象集成测试的目的

传统的集成测试有自顶向下集成和自底向上集成两种测试方式，而面向对象软件没有层次的控制结构，所以传统的自顶向下和自底向上的集成策略就没有意义。面向对象集成测试（OO Integrate Test）主要探寻导致构件交互错误的构件错误，如主要讨论以下内容：

- 哪些构件是集成测试的重点？
- 构件接口应该以什么样的顺序进行检测？
- 应该使用哪种测试设计技术检测每个接口？
- 增量式集成是最有效的技术，一次增加少数几个构件，然后测试它们的互操作性。
- 面向对象开发中的集成测试开始得早，在所有范围中都会进行，并且在每个开发增量中重复：在一个类中；在一个类层次中；在一个客户和它的服务者中；在一个相关类的簇中；在一个子系统中；在一个应用系统中。

面向对象集成测试的目的是：检测出相对独立的、单元测试无法检测出的、类相互作用时才会产生的错误，只关注于系统的结构和内部的相互作用。

15.5.2 面向对象集成测试的策略

对面向对象的集成测试（OO Integrate Test）有以下两种不同策略。

1. 基于线程的测试策略

基于线程的测试策略就是把合作对应一个输入或事件的类集合组装起来，也就是用响应系统的一个输入或一个事件的请求来组装类的集合。每个线程被集成并分别测试，应用回归测试以保证没有产生副作用。分析系统的每个输入或事件所需要的一组类，它们在一个线程中。每个线程独立地集成测试。使用协作图可对应用的类规划测试案例。

2. 基于使用的测试策略

基于使用的测试策略就是按分层来集成系统，可以先进行独立类的测试。在独立类测试之后，下一个类的层次称为从属类。从属类用独立类进行测试。这种从属类层的顺序测试直到整个系统被构造完成为止。由几乎不使用其他服务类的那些独立类开始，逐层测试与其他服务密切的层，直至全部系统；通过用例图（Use Case）捕获用户必须完成的任务。

15.5.3 面向对象集成测试的静态和动态测试

传统的集成测试，是由底向上通过集成完成的功能模块进行测试，一般可以在部分程序编译完成的情况下进行。而对于面向对象程序，相互调用的功能是散布在程序的不同类中，类通过消息相互作用申请和提供服务。类的行为与它的状态密切相关，状态不仅仅是体现在类数据成员的值上，也许还包括其他类中的状态信息。由此可见，类相互依赖极其紧密，根本无法在编译不完全的程序上对类进行测试，所以，面向对象的集成测试通常需要在整个程序编译完成后进行。此外，面向对象程序具有动态特性，程序的控制流往往无法确定，因此也只能对整个编译后的程序进行基于黑盒的集成测试。

面向对象的集成测试能够检测出相对独立的单元测试无法检测出的那些类相互作用时才会产生的错误。基于单元测试是对成员函数行为正确性的保证，集成测试只关注于系统的结构和内部的相互作用。面向对象的集成测试可以分成两步进行：先进行静态测试，再进行动态测试。

1. 面向对象的静态测试

静态测试主要针对程序的结构进行，检测程序结构是否符合设计要求。现在流行的一些测试软件都能提供一种称为“可逆性工程”的功能，即通过源程序得到类关系图和函数功能调用关系图，例如 Panorama-2 for Windows 95、Rose C++ Analyzer 等，将“可逆性工程”得到的结果与 OOD 的结果相比较，检测程序结构和实现上是否有缺陷。换句话说，通过这种方法检测 OOP 是否达到了设计要求。

2. 面向对象的动态测试

动态测试设计测试用例时，通常需要上述的功能调用结构图、类关系图或者实体关系图为参考，确定不需要被重复测试的部分，从而优化测试用例、减少测试工作量，使得进行的测试能够达到一定覆盖标准。测试所要达到的覆盖标准可以是：达到类所有的服务要求或服务提供的一定覆盖率；依据类间传递的消息，达到对所有执行线程的一定覆盖率；达到类的所有状态的一定覆盖率等。

同时也可以考虑使用现有的一些测试工具来得到程序代码执行的覆盖率。

15.5.4 面向对象集成测试的用例和测试过程

1. 面向对象集成测试的步骤

面向对象的集成测试用例可参考下列步骤：

01 先选定检测的类，参考 OOD 分析结果，仔细分析类的状态和相应的行为、类或成员函数间传递的消息、输入或输出的界定等。

02 确定覆盖标准。

03 利用结构关系图确定待测类的所有关联。

04 根据程序中类的对象构造测试用例，确认使用什么输入激发类的状态、使用类的服务和期望产生什么行为等。

值得注意的是：设计测试用例时，不但要设计确认类功能满足的输入，还应该有意识地设计一些被禁止的例子，确认类是否有不合法的行为产生，如发送与类状态不相适应的消息、要求不相适应的服务等。根据具体情况，动态的集成测试，有时也可以通过系统测试完成。

2. 集成测试设计用例的方法

集成测试的设计用例方法具有如下内容。

（1）大爆炸集成

大爆炸集成是试图通过同时测试所有的构件以论证系统的稳定性。

- 目标：通过少数测试用例运来行检测整个系统，从而论证系统的稳定性。
- 上下文和故障模型：不考虑构件之间的依赖性或风险；只有少数几个构件被加入或修改；每个构件都通过了充分的构件范围测试；无法分别对构件进行检测。
- 策略：在系统范围内应用一个测试包。

（2）自底向上集成

自底向上集成是依据使用依赖性来交错进行构件和集成测试。

- 目标：从具有最少相依性的构件开始，按照使用相依性的次序将构件加入被测系统，以证实其稳定性。
- 上下文和故障模型：具有最少相依性的构件最先测试，当这些构件通过测试后，用它们的客户对象替换驱动模块，再开始下一轮测试。
- 策略：以在相依性树中从叶子移动到根的方式进行，如果树有 n 级，则需要 n 个阶段。

（3）自顶向下集成

自顶向下集成是依据使用控制层次来交错进行构件和集成测试。

- 目标：从顶层控制对象开始，以控制层次的顺序增加构件以论证其稳定性。
- 上下文和故障模型：首先测试控制对象；增量式开发；并行硬件/软件开发；并行软件开发；框架开发。

- 策略：从最高控制层构件开始测试，以桩的形式实现服务对象；每一层按宽度优先进行，用实现代替桩，并为下面低层服务对象设置桩对象，指导系统测试完成。

（4）协作集成

协作集成是根据协作的构件和它们之间的相依性选择一个集成次序，通过一次测试一个协作组来检测接口。

- 目标：通过加入构件集合证明其稳定性，该集成被要求支持一个特定的协作。
- 上下文和故障模型：检测协作参与者之间的接口并根据协作组织集成；它们的顺序可以根据相依性、顺序激活约束或根据二者共同来选择。
- 策略：为被测系统开发相依性树；选择应用协作的序列；协作开发运行测试包。

（5）基干集成

基干集成是结合了自顶向下、自底向上和大爆炸集成，以达到一个支持迭代开发的稳定系统。

- 目标：结合自顶向下、自底向上和大爆炸集成，以验证紧密耦合的子系统之间的互操作性。
- 上下文和故障模型：系统没有基干就不能运转；基干提供了运行测试和应用必须的服务；试图设置基干的桩可能不可行或非常复杂；中层由一些簇构成，簇间耦合较松散而簇内耦合紧密。
- 策略：控制的顶层或第二层采用自顶向下测试；对底层的服务对象采用自底向上测试；基干构件采用单独测试和大爆炸测试。

（6）层次集成

层次集成用于增量式地检测一个层次体系结构中的结构和构件。

- 目标：层次集成使用增量式的方法验证一个层次体系结构中的稳定性。
- 上下文和故障模型：为一个允许只有相邻层之间接口的层次建模；通过层间的消息传播可能必须满足一个严格的、实时的期限。
- 策略：单独测试每一层；层次集成可能是自顶向下或自底向上的。

（7）客户/服务器集成

客户/服务器集成是检测一个构件松散连接的网络，其中所有构件使用一个公用的服务器构件。

- 目标：从单独测试客户和服务对象开始，使用受控增加直到所有接口被测试。
- 上下文和故障模型：通过客户/服务器集成序列达到；并行的开发和测试是可能的。
- 策略：每个客户用服务器的桩测试；该服务器用所有客户类型的桩测试，然后去掉桩进行实际测试；可以递归地应用于三重或 n 重客户/服务器体系。

（8）分布服务集成

分布服务集成是检测一个对等构件松散连接的网络。

- 目标：论证松散耦合的对等构件之间交互的稳定性。
- 上下文和故障模型：被测系统包括许多并发运行，没有专一的控制轨迹；没有专一的服务器层。

- 策略：风险驱动，从最可能有问题的接口和构件开始；反风险驱动，从最不可能有问题的接口和构件开始；相依性驱动，从依赖最少的构件开始；优先驱动，从验证高优先级性能需要的接口和构件开始。

3. 面向对象软件的集成测试过程

测试过程的第一步是进行静态测试，即针对程序的结构进行，检测程序结构是否符合设计要求。通过使用测试软件的“可逆性工程”功能，得出源程序的类系统图和函数功能调用关系图，与 OOD 结果相比较，检测程序结构和实现上是否有缺陷，检测 OOP 是否达到了设计要求。

测试过程的第二步是动态测试，即根据静态测试得出的函数功能调用关系图或类关系图作为参考，按照如下步骤设计测试用例，并达到如下测试覆盖标准。

- 设计测试用例步骤：选定检测的类，参考 OOD 分析结果，确定出类的状态和相应的行为；确定覆盖标准；利用结构关系图确定待测类的所有关联；根据程序中类的对象构造测试用例，确认使用什么输入激发类的状态，使用类的服务和期望产生什么行为等，还要设计一些类禁止的例子，确认类是否有不合法的行为产生。
- 覆盖标准：达到类所有的服务要求或服务提供的一定覆盖率；依据类间传递的消息，达到对所有执行线程的一定覆盖率；达到类的所有状态的一定覆盖率等。

15.5.5　面向对象集成测试的常见故障

面向对象集成测试的常见故障主要有如下内容：

- 整体（对象）和部件（对象）是否符合现实的关系。
- 配置/版本控制问题。
- 消息关联所指引的服务提供不正确。
- 一组子类中关于某种含义相同或基本相同的操作有不相同的接口。
- 遗漏、重叠或冲突的函数。
- 文件或数据库使用不正确或不一致的数据结构。
- 文件或数据库使用冲突的数据视图/用法。
- 破坏全局存储或数据库的数据完整性。
- 数据成员不满足数据封装的要求。
- 由于编码错误或未预料到的动态绑定导致的错误方法调用。
- 客户发送违反服务对象前提条件的消息。
- 客户发送违反服务对象的顺序约束的消息。
- 错误的对象和消息的绑定（多态目标）。
- 类不能实现要求的功能。
- 错误参数或不正确的参数值。
- 由不正确的内存管理分配/回收引起的失败。
- 不正确的使用虚拟机、ORB 或 OS 服务。
- 依据类间传递的消息，不能达到对所有执行线程的一定覆盖率。
- 目标环境的服务已经过时。

- 目标环境当前不支持试图使用的新服务。
- 构件之间的冲突。
- 资源竞争。

15.6 面向对象的系统测试

通过面向对象的单元和集成测试，仅能保证面向对象软件开发的功能得以实现，但不能确认它在实际运行时是否满足用户的需要，是否存在实际使用条件下会被诱发产生错误的隐患，需要测试它与系统其他部分配套运行的表现，以保证在系统各部分协调工作的环境下也能正常工作，这一测试为面向对象系统测试（OO System Test）。

面向对象系统测试的具体测试内容如下。

- 功能测试：测试是否满足开发要求，是否能够提供设计所描述的功能，是否用户的需求都得到满足。功能测试是系统测试最常用和必须的测试，通常还会以正式的软件说明书为测试标准。
- 强度测试：测试系统的能力最高实现限度，即软件在一些超负荷情况下的功能实现情况，如要求软件某一行为的大量重复、输入大量的数据或大数值数据、对数据库大量复杂的查询等。
- 性能测试：测试软件的运行性能。这种测试常常与强度测试结合进行，需要事先对被测软件提出性能指标，如传输连接的最长时限、传输的错误率、计算的精度、记录的精度、响应的时限和恢复时限等。
- 安全性测试：验证安装在系统内的保护机构确实能够对系统进行保护，使之不受各种干扰。安全测试时需要设计一些测试用例试图突破系统的安全保密措施，检验系统是否有安全保密的漏洞。
- 恢复测试：采用人工的干扰使软件出错，中断使用，检测系统的恢复能力，特别是通信系统。恢复测试时，应该参考性能测试的相关测试指标。
- 可用性测试：测试用户是否能够满意使用。具体体现为操作是否方便，用户界面是否友好等。

除以上测试内容外，还需要进行病毒测试、用户界面测试、容量测试、配置测试、回归测试、确认测试、容错性测试、可靠性测试、易用性测试、安装/卸载测试等。

15.7 面向对象软件的测试用例设计

15.7.1 面向对象软件的测试用例设计原则

面向对象软件测试用例着眼于适当的操作序列，以实现对类的说明。面向对象软件的测试用例设计原则主要有如下 4 点内容：

- 关注于设计合适的操作序列以测试类的状态。
- 对每个测试用例应当给予特殊的标识，并且还应当与测试的类有明确的联系。

- 测试目的应当明确。
- 应当为每个测试用例开发一个测试步骤列表，列表包含以下内容：列出所要测试对象的说明；列出将要作为测试结果的消息和操作；列出测试对象可能发生的例外情况；列出外部条件，为了正确对软件进行测试所必须有的外部环境的变化；列出为了帮助理解和实现测试所需要的附加信息。

15.7.2　面向对象软件的测试用例设计方法

1. 面向对象软件测试用例设计的要求

每个测试用例应被惟一标识，并被显式地和将被测试的类相关联；测试的目的应被陈述；对每个测试应开发一组测试步骤，应包含如下内容：

- 被测试的对象的一组特定状态。
- 被作为测试的结果使用的一组消息和操作。
- 当对象被测试时可能产生的一组异常。
- 一组外部条件。
- 将辅助理解或实现测试的补充信息。

2. 面向对象软件的测试用例设计方法

（1）基于故障的测试

在面向对象软件测试中，由于系统必须满足用户的需求，因此，基于故障的测试要从分析模型开始，考察可能发生的故障。为了确定这些故障是否存在，可设计用例去执行设计或代码。基于故障测试的重点如下：

- 具有较高的发现可能故障的能力。
- 从分析模型开始，考察可能发生的故障，设计用例去执行设计和代码。
- 发现消息联系中“可能的故障”。“可能的故障”一般为意料之外的结果、错误地使用了操作/消息、不正确引用等。为了确定由操作（功能）引起的可能故障必须检查操作的行为。
- 除用于操作测试外，还可用于属性测试，用以确定其对于不同类型的对象行为是否赋予了正确的属性值。
- 从客户对象（主动）上发现错误。
- 基于故障的测试发现不正确的规格说明、用户不需要的功能或缺少用户需要的功能。
- 发现没有考虑的子系统间的交互作用。

基于故障测试的关键取决于测试设计者如何理解“可能的故障”。而在实际中，要求设计者做到这点是不可能的。

故障测试方法除用于操作测试外，还可用于属性测试，用以确定其对于不同类型的对象行为是否赋予了正确的属性值。因为一个对象的“属性”是由其赋予属性的值定义的。

（2）基于场景（脚本）的测试

基于场景的测试关心用户做什么，而不是产品做什么。它意味着捕获用户必须完成的任务，

然后在测试时应用它们或它们的变体。它往往在单个测试中处理多个子系统。

这种基于场景的测试有助于在一个单元测试情况下检查多重系统，所以基于场景测试用例的测试比基于故障测试不仅更实际（接近用户），而且更复杂一点。

例如：考察一个文本编辑器的打印场景测试的用例设计。

Use Case1：打印最终的文档。

测试案例：

- 打印完整的文档。
- 在文档中修改某些页的内容。
- 再次打印完整文档或某些页。

Use Case2：打印复制的新页。

测试案例：

- 打开文档。
- 选择菜单中的 print 选项。
- 设定打印（还是完整文档）。
- 单击 print 开始打印。
- 关闭文档。

其执行事件序列是：打印整个文件；移动文件，修改某些页；当某页被修改，就打印某页；有时要打印许多页。

显然，测试者希望发现打印和编辑两个软件功能是否能够相互依赖，否则就会产生错误。

（3）面向对象类的随机测试

如果一个类有多个操作（功能），这些操作（功能）序列有多种排列。而这种不变化的操作序列可随机产生，用这种可随机排列的序列来检查不同类实例的生存史，就称为随机测试。例如：银行单机 ATM 自动取款账户 account 类上的所有操作：open、setup、deposit、withdraw、balance、summarize、creditlimit、close。

账户的最小生命历史应该包括的操作如下：

open.setup.deposit.withdraw.close

账户的其他行为都包含在下列操作中（最大包容）：

open.setup.deposit.[deposit|withdraw|balance|summarize|creditlimit].withdraw. close

这样可随机地产生测试账户类的案例。

- 测试案例 1：open.setup.deposit.balance.summarize.withdraw.close。
- 测试案例 2：open.setup.deposit.withdraw.balance.creditlimit.withdraw.close。

（4）类层次的分割测试

类层次的分割测试可以减少利用完全相同的方式检查类测试用例的数目，类似于等价类划分。类层次的分割测试可分为基于状态的分割、基于属性的分割、基于类型的分割。

① 基于状态的分割

基于状态的分割是指按类操作是否改变类的状态来分割（归类）。这里仍以 account 类为例，改变状态的操作有 deposit、withdraw，不改变状态的操作有 balance、summarize、creditlimit。如果测试按检查类操作是否改变类的状态来设计，则结果如下。

用例 1：执行操作改变状态。

open+setup+deposit+deposit+withdraw+withdraw+close

用例 2：执行操作不改变状态。

open+setup+deposit+summarize+creditlimit+withdraw+close

② 基于属性的分割

基于属性的分割是指按类操作所用到的属性来分割（归类），如果仍以一个 account 类为例，其属性 creditlimit 能被分割为三种操作：用 creditlimit 的操作、修改 creditlimit 的操作、不用也不修改 creditlimit 的操作。这样，测试序列就可按每种分割来设计。

③ 基于类型的分割

基于类型的分割是指按完成的功能分割（归类），例如，在 account 类的操作中，可以分割为：初始操作 open、setup；计算操作 deposit、withdraw；查询操作 balance、summarize、creditlimit；终止操作 close。

（5）状态机模型导出的测试

状态机是一个系统，它的输出是由当前的输入和过去的输入决定的。状态机模型导出的测试由状态、转换、事件和动作构成。状态机的机制由如下 5 点构成：

- 机器开始于初态。
- 机器在一个不确定的时间间隔内等待一个事件。
- 如果一个事件在当前状态没有被接受，则被忽略。
- 如果事件在当前状态被接受，则说明指定的转换被触发。
- 从第二步开始重复这个周期，直到结果状态变成终止状态。

状态机的测试设计主要是：实现必须符合指定的状态机的约束；了解实现如何与观察到的约束不相符合，为测试设计提供了重点，也为评估测试设计策略提供了基准。

一个控制错误允许一个不正确的事件序列被接受或产生一个不正确的输出动作序列。

控制错误的类型主要有如下几点：

- 丢失的或不正确的转换。
- 丢失的或不正确的事件。
- 丢失的或不正确的动作。
- 多余的、丢失的或讹误状态。
- 潜行路径（一个不应被接受的消息被接受）。
- 非法消息的失效（一个不期望的消息引起的失效）。
- 陷阱门（实现了没有定义的消息）。

状态模型检查表的主要内容包括：结构、状态名称、受监视转换、好的子类行为、健壮性等。模型在结构上是完全且一致的，具有如下重点：

- 一个状态被指定为初态并且只有一个离开的转换。
- 至少有一个终止状态并且只有一个进入转换。
- 没有等价状态。
- 每一个状态都可以从初态到达。
- 从任何一个状态都可以到达终止状态。
- 每一个被定义的事件和动作至少出现在一个转换上。
- 除了初态和终态外，每个状态至少有一个输入转换和输出转换。
- 对任一给定状态，同样的事件不能出现在多个转换上。
- 状态机是完全确定的。

不好的状态名经常是不完全和不正确设计的征兆，设计状态名称时的主要内容如下：

- 状态名称在应用领域中是有意义的。
- 当怀疑一个状态名时，就把它删掉。
- 不要用“等待的 x”或“等待 x”作为状态名。
- 状态名没有表达不相关的信息，没有名称描述和动作相关的过程；没有名称用于输入描述。

对于条件转换的主要注意事项如下：

- 在状态转换的条件集中，条件表达式产生的真值的全部范围必须被覆盖。
- 对一个转换，每一个条件都和其他的条件相互排斥。
- 一个抽象状态模型的条件变量是被测类的客户提供的输入。
- 一个具体类状态模型的条件变量是由被测类的实现定义的——例如 self 或 this。
- 条件表达式的计算对被测类不会产生任何副作用。

好的子类的行为是指以下情况：

- 子类没有删除任何超类状态，在超类中接受的转换也被子类接受。
- 子类不会减弱任何超类状态的状态不变量。
- 子类不会破坏任何超类状态的状态不变量。
- 子类或者加强超类状态不变量，也可能仅仅继承它。
- 子类可以添加由子类引入的实例变量定义的正交状态。
- 超类转换中所有的条件和子类转换相同或弱于子类。
- 所有被继承的输出动作和子类的责任一致。
- 发送到对象的消息不会对被测类产生副作用。

模型规定了如下健壮行为：

- 对没有被明确拒绝的事件，有一个明确规定的错误处理和异常处理机制。
- 对于一个非法消息的结果，实现的状态不会被错误地改变、被讹误或没有被定义。
- 对每一个动作，没有与这个动作相关的结果状态的副作用或异常。
- 对前置/后置条件、不变式的冲突，规定了明确的异常、错误注册和恢复机制。

- 对丢失或延迟的事件规定了明确的异常、错误注册和恢复机制。
- 对每一个超时，规定了明确的异常、错误注册和恢复机制。
- 被测系统的目标环境的主要和不重要的失败模式被分类，对每个失败模式，规定了一个明确的异常机制。
- 对每一失败模式，指定了结果状态或硬性的终止状态。
- 重新开始机制被建模并定义了一个到正常状态的路径。

（6）面向 UML 的测试

统一建模语言 UML 具有定义良好、易于表达、功能强大的特点，不仅支持面向对象的分析与设计，而且支持从需求分析开始的软件开发的全过程。UML 的目标是以面向对象的方式来描述任何类型的系统，它提供了非常丰富的图例模型。

① 用例图的测试需求

用例图的测试需求如表 15-2 所示。

表 15-2 用例图的测试需求

关系	测试需求（至少有一个测试用例检查）
执行者和用例通信	每一用例和每一执行者的用例
用例 1 扩展用例 2	每一个完全的扩展组合
用例 1 使用用例 2	每一个完全的使用组合

② 类图的测试需求

类图的测试需求如表 15-3 所示。

表 15-3 类图的测试需求

关系	测试需求（至少有一个测试用例检查）
关联	关联关系的特定应用
聚集	类和构件的独立创建和析构
构成	类和构件的顺序创建和析构、类和构件失败的独立创建和析构
泛化	在每一个子类中，每一个超类的特性被使用
依赖	依赖关系的特定应用
精化	精化类中被精化类型的每一行为

③ 顺序图的测试需求

顺序图的测试需求如表 15-4 所示。

表 15-4 顺序图的测试需求

关系	测试需求（至少有一个测试用例检查）
客户调用服务者（同步）	客户调用服务者并返回
客户调用服务者（异步）	客户调用服务者并返回，服务者继续执行
客户调用服务者 1、2…	客户调用服务者 1、2…
客户重复到服务者的调用	客户重复到服务者的调用
客户递归调用服务者	客户递归调用服务者

④ 活动图的测试需求

活动图的测试需求如表 15-5 所示。

表 15-5　活动图的测试需求

关系	测试需求（至少有一个测试用例检查）
动作 1 在动作 2 之前	动作 2 跟随动作 1
动作依赖于同步点	动作跟随同步点
动作 2、3…跟随动作 1	动作 2 跟随动作 1、动作 3 跟随动作 1…
动作依赖信号	在信号到达之后执行动作
同步点跟随动作	在动作之后到达同步点

⑤ 协作图的测试需求

协作图的测试需求如表 15-6 所示。

表 15-6　协作图的测试需求

关系	测试需求（至少有一个测试用例检查）
消息 1 在消息 2 之前	消息 1 和消息 2
客户发送消息到服务者	客户发送消息到服务者并返回
客户也许发送消息到服务者（条件调用）	客户发送消息到服务者并返回、客户不发送消息到服务者
客户重复到服务者的消息	客户重复到服务者的消息并返回
向自己发送递归消息	客户递归地调用自己
客户到服务者的异步调用	客户发送消息到服务者并返回，服务者收到消息

⑥ 构件图的测试需求

构件图的测试需求如表 15-7 所示。

表 15-7　构件图的测试需求

关系	测试需求（至少有一个测试用例检查）
构件发送到接口的消息	客户发送消息到服务者并返回
构件依赖构件	特定应用

⑦ 配置图的测试需求

配置图的测试需求如表 15-8 所示。

表 15-8　配置图的测试需求

关系	测试需求（至少有一个测试用例检查）
构件运行在节点上	在每一个指定的主机节点上，构件能被加载并且运行
节点和节点的通信	打开、传送、关闭到每一个远程构件的通信

第16章　软件缺陷测试和测试评估

在程序中存在的软件缺陷（如文档缺陷、代码缺陷、测试缺陷、过程缺陷）导致系统或部件不能实现其功能，引起系统的失效。对软件缺陷测试和测试评估是不可缺少、相当重要的。本章重点讨论以下内容：

- 软件缺陷概述。
- 软件缺陷的生命周期。
- 软件缺陷的跟踪管理。
- 软件测试的评估。

16.1　软件缺陷概述

16.1.1　软件缺陷的定义

缺陷（Bug）是指程序中存在的错误，如语法错误、拼写错误或者一个不正确的程序语句，缺陷可能出现在设计中，甚至在需求、规格说明或其他文档中。软件缺陷导致系统或部件不能实现其功能，引起系统的失效。缺陷定义为：

- 软件没有达到产品说明书表明的功能。
- 程序中存在语法错误。
- 程序中存在拼写错误。
- 程序中存在不正确的程序语句。
- 软件出现了产品说明书中不一致的表现。
- 软件功能超出产品说明书的范围。
- 软件没有达到用户期望的目标。
- 测试员或用户认为软件的易用性差。

按照定义，可以将缺陷分为文档缺陷、代码缺陷、测试缺陷、过程缺陷。

- 文档缺陷：文档缺陷是指对文档的静态检查过程中发现的缺陷，通过测试需求分析、文档审查对被分析或被审查的文档发现的缺陷。
- 代码缺陷：代码缺陷是指对代码进行同行评审、审计或代码走查过程中发现的缺陷。
- 测试缺陷：测试缺陷是指由测试执行活动发现的被测对象（被测对象一般是指可运行的代码、系统，不包括静态测试发现的问题）的缺陷，测试活动主要包括内部测试、连接测试、系统集成测试、用户验收测试，测试活动中发现的缺陷为测试缺陷。
- 过程缺陷：过程缺陷是指通过过程审计、过程分析、管理评审、质量评估、质量审核等活动发现的关于过程的缺陷和问题。过程缺陷的发现者一般是质量经理、测试经理、管理人员等。

16.1.2 软件缺陷的特征

软件缺陷的特征主要有如下 7 点内容。

- 单一准确：每个报告只针对一个软件缺陷。在一个报告中报告多个软件缺陷的弊端常常会导致缺陷部分被注意和修复，不能得到彻底修正。
- 可以重现（要求软件缺陷具有精确的步骤）：提供缺陷的精确操作步骤，容易重现这个缺陷。
- 完整统一：提供完整、前后统一的软件缺陷的步骤和信息，如图片信息、Log 文件等。
- 短小简练：通过使用关键词，使软件缺陷的标题的描述简练，准确解释产生缺陷的现象，如“主页”、“导航栏”、“分辨率”等关键词。
- 特定条件：软件功能在通常情况下没有问题，而是在某种特定条件下会存在缺陷，所以软件缺陷描述不要忽视特定条件，如特定的操作系统、浏览器或某种设置等。
- 补充完整：测试人员发现 Bug 要保证它被正确的报告，并且得到应有的重视，继续监视其修复的全过程。
- 不做评价：在软件缺陷描述不要带有个人观点，对开发人员进行评价。软件缺陷报告是针对产品、针对问题本身，将事实或现象客观地描述出来就可以，不需要任何评价或议论。

16.1.3 软件缺陷的类型

软件缺陷的类型可分为：需求缺陷、设计缺陷、结构缺陷、系统结构缺陷、测试设计与测试执行错误、功能类缺陷、性能类缺陷、系统模块接口类缺陷、用户界面类缺陷、数据处理类缺陷、流程类缺陷、提示信息类缺陷、软件包类缺陷、建议类缺陷、常识类缺陷、文档缺陷。软件缺陷类型如表 16-1 所示。

表 16-1 软件缺陷类型

缺陷类型	描述
需求缺陷	需求有误、需求逻辑错误、需求不完备、需求文档描述问题、需求更改
设计缺陷	功能的使用对用户带来不便及不符合行业标准的问题：设计不合理、设计文档描述的问题、设计变更带来的问题
结构缺陷	控制流和控制顺序错、处理错
系统结构缺陷	操作系统引用或使用错误、软件结构错误、恢复错误、执行错误、诊断错误、分割覆盖错误、引用环境错误
测试设计与测试执行错误	测试设计错误、测试执行错误、测试文档有误、测试用例不充分、其他测试错误
功能类缺陷	影响了各种系统功能或逻辑的缺陷、重复的功能、多余的功能、功能实现与设计要求不相符、功能使用性（方便性、易用性）不够
性能类缺陷	不满足系统可测量的属性值，如事务处理速率、并发量、数据量、压缩率、响应时间
系统模块接口类缺陷	与其他组件、模块、设备驱动程序、调用参数、控制块或参数列表等不匹配、冲突
用户界面类缺陷	影响了用户界面、人机交互特性；屏幕格式、用户输入灵活性、结果输出格式等方面的缺陷；界面不美观；控件排列、格式不统一；焦点控制不合理或不全面
数据处理类缺陷	数据有效性检测不合理、数据来源不正确、数据处理过程不正确

（续表）

缺陷类型	描述
流程类缺陷	流程控制不符合要求、流程实现不完整
提示信息类缺陷	提示信息重复或出现时机不合理、提示信息格式不符合要求、提示框返回后焦点停留位置不合理
软件包类缺陷	由于软件配置库、变更管理或版本控制引起的错误
建议类缺陷	功能性建议、操作建议、检校建议、说明建议
常识类缺陷	违背正常习俗习惯，如日期、节日等
文档缺陷	影响发布和维护，包括注释、用户手册、设计文档

16.1.4 Bug 状态

缺陷状态是指缺陷通过一个跟踪修复过程的进展情况。Bug 状态可分为如下几种。

- 激活或打开（Active or Open）：问题还没有解决，存在源代码中，确认“提交的缺陷”，等待处理，如新报的缺陷。
- 已提交：测试员发现 Bug 后提交到 Bug 管理系统中的状态（初始状态）。
- 已修改（Fixed or Resolved）：已被程序员检查、修复过的缺陷，通过单元测试，认为已解决但还没有被测试人员验证提交到 Bug 管理系统中的状态。
- 不修改（保留）：程序员或项目经理根据需求分析、概要设计、详细设计说明书等上的要求经过考虑后决定对 Bug 不进行修改（由于技术原因或第三者软件的缺陷，开发人员不能修复的缺陷），其 Bug 的状态为不修改。
- 延迟：根据目前项目进程或计划等情况，暂时延期的状态，缺陷可以在下一个版本中解决。
- 待讨论：需要进行讨论后才能决定是否需要修改 Bug 的状态。
- 已验证：已经解决的并经过测试员复测的 Bug 状态。
- 关闭：完全解决了，只供以后备查的状态。
- 重新打开：测试人员验证后还依然存在的缺陷，等待开发人员进一步修复，重新打开以前关闭的 Bug 状态（在 Bug 工具中，可以自己定制适合项目的状态项目，如废除、拒绝等）。

16.1.5 Bug 的等级划分与优先级

1. Bug 的等级划分

Bug 的等级可以划分为 4 级。

- 严重：死机、数据丢失、主要功能完全丧失、用户数据受到破坏、系统崩溃等错误。修改优先级为最高，该级别需要程序员立即修改。
- 较高：系统的主要功能部分丧失、数据不能保存，导致严重的问题或致命的错误。修改优先级为较高，该级别需要程序员尽快修改。
- 一般：系统的部分功能没有完全实现，但不影响用户的正常使用，如提示信息不太准确或用户界面差、操作时间长等一些问题。修改优先级为中，该级别需要程序员修改。
- 轻微：微小的问题，对功能几乎没有影响，产品及属性仍可使用，如有个错别字、操作者不方便。修改优先级为低，该级别需要程序员修改或不修改。

2. Bug 的优先级

Bug 的优先级一般与 Bug 等级挂钩，可以分为 4 级。

- 1 级（严重）：立即解决。缺陷导致系统几乎不能使用或测试不能继续，需要立即修复。
- 2 级（较高）：缺陷严重，影响测试，需要优先考虑。
- 3 级（一般）：正常排队缺陷需要正常排队等待修复。
- 4 级（轻微）：缺陷可以在开发人员有时间的时候被纠正。

16.1.6 软件缺陷的标识、种类和属性

1. 缺陷标识

缺陷标识是指标记某个缺陷的惟一表示，可以使用数字序号表示，如表 16-2 所示。

缺陷标识是按照问题的复杂度来排列的，类型 10～40 是比较简单的编码缺陷，类型 50～100 是比较复杂的设计缺陷。

表 16-2 缺陷标识

缺陷标识编号	缺陷类型名称	描述
10	文档	注释、消息需求、设计类文档
20	语法	拼写、标点、打字、指令格式
30	赋值	如声明、重复命名、作用域
40	接口	过程调用、输入/输出、用户格式与其他组件、模块或设备驱动程序、调用参数、控制块或参数列表相互影响的缺陷
50	打包	由于配置库、变更管理或版本控制引起的错误
60	数据、函数	结构、内容、逻辑、指针、循环、递归、计算、函数缺陷
70	用户接口	人机交互特性、屏幕格式、确认用户输入、功能有效性
80	性能	不满足系统可测量的属性值，如执行时间、事务处理速率等
90	标准	不符合各种标准的要求，如编码标准、设计符号等
100	系统、环境	设计、编译、配置、计时、内存、其他支持系统的问题

2. 缺陷种类

软件缺陷的种类有如下 15 点内容：

- 功能不正常。
- 软件在使用上不方便。
- 软件的结构未做良好规划。
- 功能不充分。
- 与软件操作者的互动不良。
- 使用性能不佳。
- 未做好错误处理。
- 边界错误。
- 计算错误。
- 使用一段时间所产生的错误。

- 控制流程的错误。
- 在大数据量压力之下所产生的错误。
- 在不同硬件环境下产生的错误。
- 版本控制不良所产生的错误。
- 软件文档的错误。

3. 软件缺陷的属性

软件缺陷的属性主要有如下 10 点内容：

- 缺陷标识。
- 缺陷描述与缺陷注释。
- 缺陷类型。
- 缺陷严重程度。
- 缺陷产生的可能性。
- 缺陷的优先级。
- 缺陷状态。
- 软件缺陷的起源。
- 软件缺陷的来源。
- 软件缺陷的根源。

软件缺陷的主要属性操作如表 16-3 所示。

表 16-3　软件缺陷属性操作

缺陷属性	软件测试	同行评审
缺陷标识（Identifier）	需要记录	需要记录
缺陷类型（Type）	需要记录	需要记录
缺陷严重程度（Severity）	需要记录	需要记录
解决优先级（Priority）	需要记录	不考虑
缺陷状态（Status）	需要记录	不考虑
缺陷起源（Origin）	需要记录	需要记录
缺陷原因（Cause）	可以不考虑/可以记录	可以不考虑/可以记录

16.1.7　缺陷的起源、来源和根源

1. 缺陷的起源

缺陷起源是指缺陷引起的故障或事件第一次被检测到的阶段，给软件带来缺陷的原因很多，如需求、构架、设计、编码、测试、用户等。

- 需求：参与人员的过度自信，在需求阶段产生的错误。
- 构架：人员之间的沟通交流不够，交流上有误解或者根本不进行交流，在系统构架设计阶段产生的错误。
- 设计：工期短，任务重，时间压力大，在程序设计阶段产生的错误。
- 编码：在编码阶段程序设计本身由错误产生的错误。

- 测试：在测试阶段发现的缺陷。
- 用户：在用户使用阶段发现的错误。

2. 缺陷的来源

缺陷的来源是指缺陷所在的地方，如需求说明书、设计文档、系统集成接口、数据流（库）、程序代码等。

- 需求说明书：需求说明书的错误或由不清楚引起的问题。
- 设计文档：设计文档描述不准确，以及与需求说明书不一致的问题。
- 系统集成接口：由于系统模块参数不匹配、开发组之间缺乏协调引起的缺陷。
- 数据流（库）：由于数据字典、数据库中的错误引起的缺陷。
- 程序代码：纯粹是由编码中的问题所引起的缺陷。

3. 缺陷的根源

缺陷的根源是指造成软件错误的根本因素，如测试策略、过程工具和方法、团队/人、缺乏组织和通信、硬件、软件、工作环境等。

- 测试策略：错误的测试范围、误解测试目标、超越测试能力等。
- 过程工具和方法：无效的需求收集过程、过时的风险管理过程、不使用的项目管理方法，没有估算规程、无效的变更控制过程等。
- 团队/人：项目团队职责交叉、缺乏培训、没有经验的项目团队、缺乏士气和动机不纯等。
- 缺乏组织和通信：缺乏用户参与、职责不明确、管理失败等。
- 硬件：硬件配置不对或缺乏、处理器缺陷导致算术精度丢失、内存溢出等。
- 软件：软件设置不对或缺乏、操作系统错误导致无法释放资源、工具软件的错误、编译器的错误等。
- 工作环境：组织机构调整、预算改变、工作环境恶劣。

16.1.8 Bug 记录

Bug 记录的内容如表 16-4 所示。

表 16-4 Bug 记录内容

编号		测试日期	
标题			
项目模块		测试阶段	
测试员		操作环境	
Bug 类型		等级及优先级	
详细描述			
步骤（操作、数据输入等）： 结果： 期望： 备注：			
程序员		解决日期	
解决方案			

16.2　软件缺陷的生命周期

软件缺陷的生命周期是指一个软件缺陷被发现、报告到这个缺陷被修改、验证直至最后关闭的完整过程。软件缺陷的生命周期分为简单的软件缺陷生命周期和复杂的软件缺陷生命周期。

1. 简单的软件缺陷生命周期

简单的软件缺陷生命周期的过程如下。

- 发现→打开：测试人员找到软件缺陷并将软件缺陷提交给开发人员。
- 打开→修复：开发人员重现、修改缺陷，然后提交给测试人员去验证。
- 修复→关闭：测试人员验证修改过的软件，关闭已不存在的缺陷。

“发现→打开→修复→关闭”是一种理想的状态，在实际的工作中是很难有这样顺利的，需要考虑到各种突发情况。

2. 复杂的软件缺陷生命周期

复杂的软件缺陷生命周期如下。

- 新建一个软件缺陷，这个软件缺陷是（Open）状态，进行 Bug 审查，不是代码问题就是设计需要修改。
- 新建一个软件缺陷，这个软件缺陷是（Open）状态，进行 Bug 审查，以后修改的就可以延期。
- 新建一个软件缺陷，这个软件缺陷是（Open）状态，进行 Bug 审查，实际没有这个 Bug，可以将其关闭。
- 新建一个软件缺陷，这个软件缺陷是（Open）状态，看是否可重现，如果不能重现，就是缺少信息，需要返回到（Open）状态；如果能够重现就进行修正，修正后关闭，进行回归测试。

软件缺陷生命周期中的不同阶段是测试人员、开发人员和管理人员一起参与、协同测试的过程。软件缺陷一旦发现，便进入测试人员、开发人员、管理人员的严格监控之中，直至软件缺陷的生命周期终结，这样可保证在较短的时间内高效率地关闭所有缺陷、缩短软件测试的进程、提高软件质量，同时减少开发和维护成本。

每一个软件缺陷都有 6 个生命状态：打开（Open）、缺陷修改（Working）、缺陷验证（Verify）、缺陷删除（Cancel）、缺陷关闭（Close）、缺陷延期（Defer）。它们的基本定义如下。

- Open 态：缺陷初试状态，测试员报告一个缺陷，缺陷生命周期开始。
- Working 态：缺陷修改状态，程序员接收缺陷，正在修改中。
- Verify 态：缺陷验证状态，程序员修改完毕，等待测试员验证。
- Close 态：缺陷关闭状态，测试员确认缺陷被改正，将缺陷关闭。
- Cancel 态：缺陷删除状态，测试员确认不是缺陷，将缺陷置为删除状态（不做物理删除）。
- Defer 态：缺陷延期状态，管理者确认缺陷需要延期修改或追踪，将缺陷置为延期状态。

上述打开态、缺陷修改态、缺陷验证态，称为缺陷的活动态；缺陷关闭态、缺陷删除态、缺陷延期态，称为缺陷的终结态。

软件缺陷的生命周期示意图如图 16-1 所示。

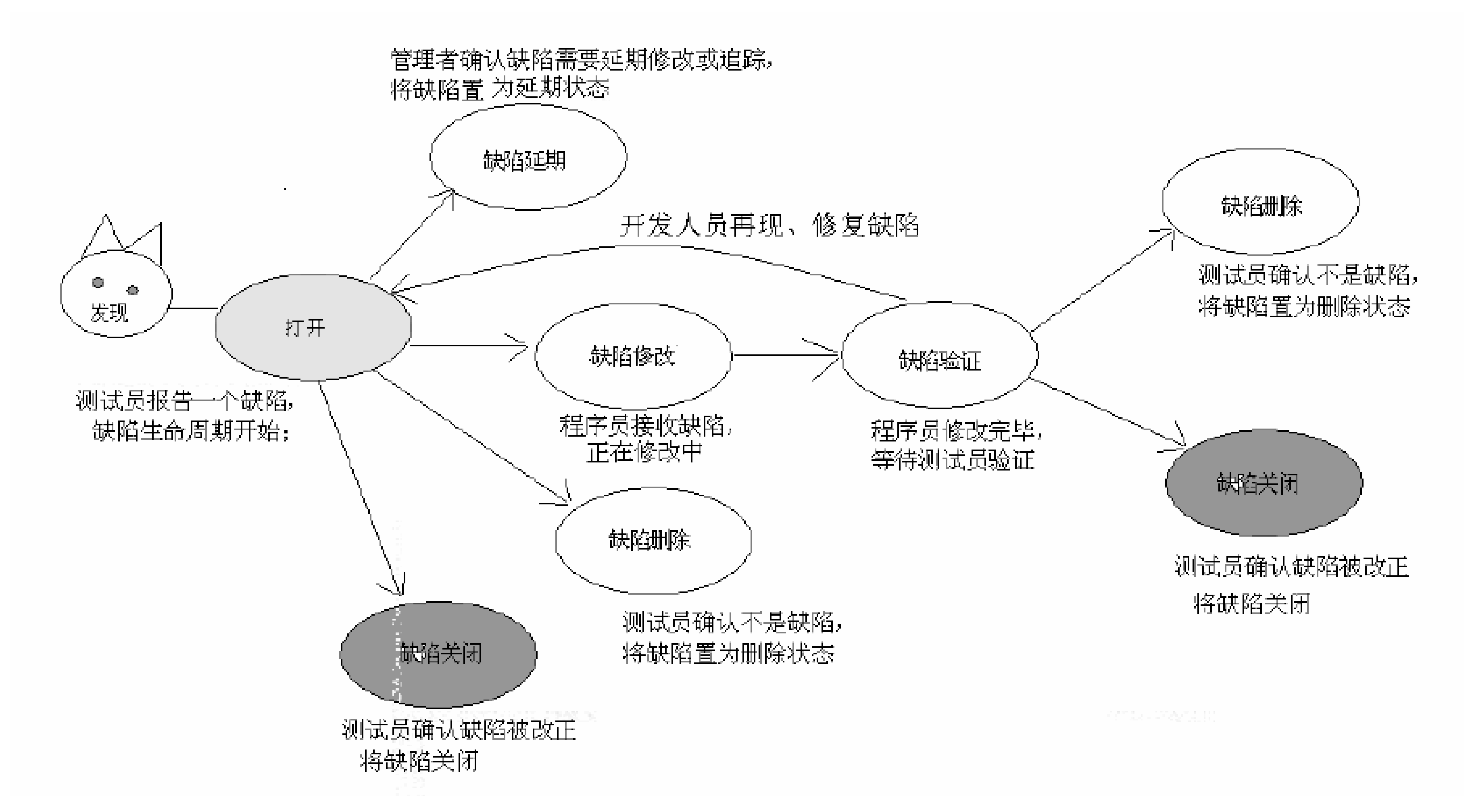


图 16-1 软件缺陷的生命周期示意图

16.3 软件缺陷的跟踪管理

16.3.1 软件缺陷的测试报告

软件中不可能没有缺陷，发现很多的缺陷对于测试工作来说是件很正常的事。如果测试中发现缺陷，则需要进行报告。

1. 报告软件缺陷的原则

报告软件缺陷的原则是：尽快报告软件缺陷，并有效地描述软件缺陷，在报告软件缺陷时不做任何评价。其中有效的软件缺陷描述要求如下：

- 简单与短小。
- 明确指明错误类型。
- 单一。
- 使用 IT 业界惯用的表达术语和表达方法。

2. 软件缺陷报告的信息

任何一个缺陷跟踪系统的核心都是“软件缺陷报告”，软件缺陷报告的详细信息如表 16-5 所示。

表 16-5　软件缺陷报告的详细信息

分类	项目	描述
可跟踪信息	缺陷 ID	惟一的、自动产生的缺陷 ID，用于识别、跟踪、查询
软件缺陷的基本信息	缺陷状态	可分为“打开或激活的”、“已修正”、“关闭”等
	缺陷标题	描述缺陷的最主要信息
	缺陷的严重程度	一般分为“致命”、“严重”、“一般”、“较小”等 4 种程度
	缺陷的优先级	描述处理缺陷的紧急程度，如优先级最高的等级、正常的等级、优先级最低的等级
	缺陷的产生频率	描述缺陷发生的可能性 1%～100%
	缺陷提交人	缺陷提交人的名字（会和邮件地址联系起来），一般就是发现缺陷的测试人员或其他人员
	缺陷提交时间	缺陷提交的时间
	缺陷所属项目/模块	缺陷所属的项目和模块，最好能较精确地定位至模块
	缺陷指定解决人	估计修复这个缺陷的开发人员，在缺陷状态下由开发组长指定相关的开发人员；也会自动和该开发人员的邮件地址联系起来，并自动发出邮件
	缺陷指定解决时间	开发管理员指定的开发人员修改此缺陷的时间
	缺陷验证人	验证缺陷是否有真正被修复的测试人员；也会和邮件地址联系起来
	缺陷验证结果描述	对验证结果的描述（通过、不通过）
	缺陷验证时间	对缺陷验证的时间
软件缺陷的详细描述	步骤	对缺陷的操作过程，按照步骤一步一步地描述
	期望的结果	按照设计规格说明书或用户需求，在上述步骤之后，得到所期望的结果，即正确的结果
	实际发生的结果	程序或系统实际发生的结果，即错误的结果
测试环境说明	测试环境	对测试环境的描述，包括操作系统、浏览器、网络带宽、通信协议等
必要的附件	图片、Log 文件	对于某些文字很难表达清楚的缺陷，使用图片等附件是必要的；对于软件崩溃现象，需要使用 Soft_ICE 工具去捕捉日志文件作为附件提供给开发人员

3. 软件缺陷的手工报告记录

显然，在软件测试工作中，每个测试用例的结果都必须进行记录。如果使用软件缺陷跟踪系统，那么测试工具将自动记录软件缺陷的相关信息。如果测试是采用手工记录和跟踪软件缺陷，那么有关软件缺陷的信息可以直接记录在相应的文档中，文档如表 16-6 所示。

表 16-6　采用手工记录和跟踪软件缺陷信息表

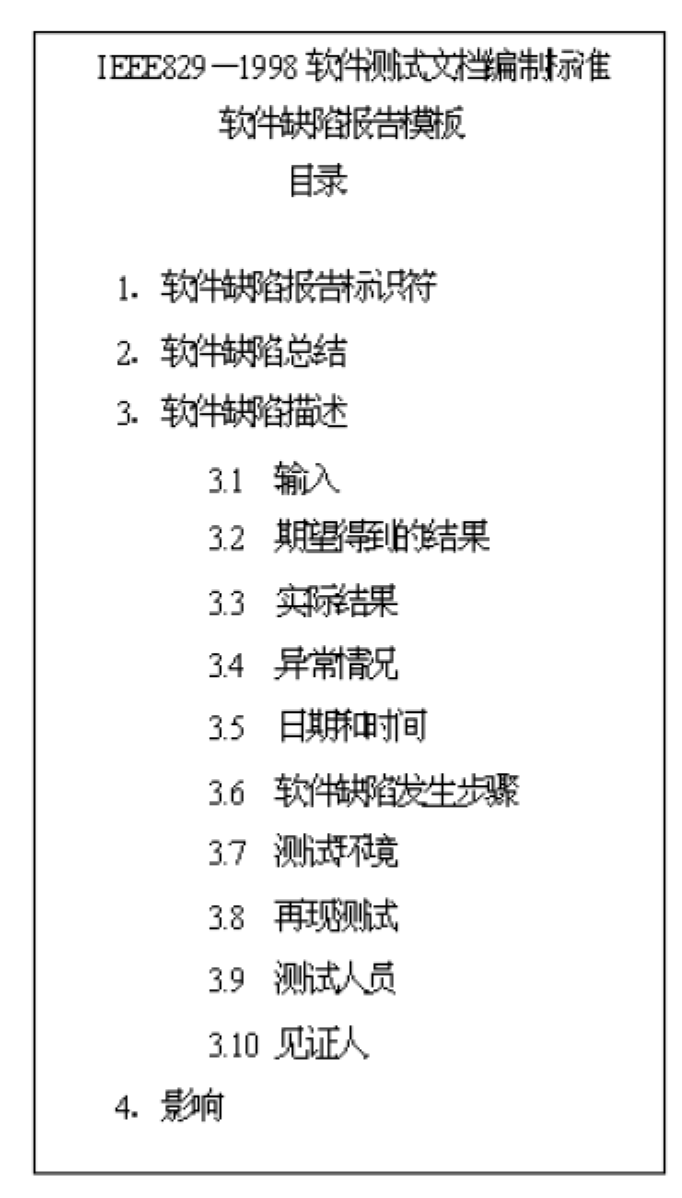
公司名称		Bug 报告		Bug#		
软件		版本				
测试员		日期				
严重性		优先级		是否会重现		是 否
标题						

（续表）

描述：					
解决方法：					
解决日期		解决人		版本号	
解决描述：					
重新测试人		测试版本号		测试日期	
重新测试描述：					
签名					
策划		测试环境			
编程		项目管理			
销售		技术支持			

4. IEEE 软件缺陷的报告模板

ANS/IEEE 829-1998 标准定义了一个称为软件缺陷报告的文档，用于报告“在测试期间发生的任何异常事件”。模板标准如图 16-2 所示。



IEEE829－1998 软件测试文档编制标准
软件缺陷报告模板
目录

1. 软件缺陷报告标识符
2. 软件缺陷总结
3. 软件缺陷描述
    3.1 输入
    3.2 期望得到的结果
    3.3 实际结果
    3.4 异常情况
    3.5 日期和时间
    3.6 软件缺陷发生步骤
    3.7 测试环境
    3.8 再现测试
    3.9 测试人员
    3.10 见证人
4. 影响

图 16-2　IEEE 软件缺陷报告标准模板

16.3.2　软件缺陷的分离和重现

软件缺陷的分离和重现是考验测试人员的专业技能，测试人员要想有效报告软件缺陷，就要对软件缺陷以明显、通用和重现的形式进行描述。

1. 缺陷的分离

缺陷分离的方法主要有以下几种：

- 确保所有的测试步骤都被记录（记录每一个测试步骤、每一件工作）。
- 注意时间和运行条件上的因素（查找时间依赖和竞争条件的问题）。
- 注意软件的边界条件、内存容量和数据溢出的问题。
- 注意事件发生次序导致的软件缺陷。
- 考虑资源依赖性和内存、网络、硬件共享的相互作用。
- 注意系统的压力条件。
- 不能忽视硬件，与软件不同，硬件不按预定的方式工作。

2. 缺陷的重现

缺陷的重现要采取繁杂的步骤才能实现，或者根本无法重现。开发人员有时可以根据相对简单的错误信息就能找出问题所在。因为开发人员熟悉代码，因此看到症状、测试用例步骤和分离问题的过程时，可能得到查找软件缺陷的线索。一个软件缺陷的重现问题有时需要小组的共同努力。如果软件测试人员尽最大努力分离软件缺陷，也无法表达准确的重现步骤，那么仍然需要记录和报告软件缺陷。

缺陷重现的方法与缺陷分离的方法相同，这里不再赘述。

16.3.3 软件缺陷的跟踪系统

软件缺陷跟踪系统（Bug Tracking System）是用于记录、跟踪、归并类处理软件开发过程出现的 Bug 和硬件系统中存在的缺陷（Defect）。集中管理软件测试过程中所发现缺陷的数据库程序，可以通过添加、修改、排序、查找、存储操作来管理软件缺陷。目的是保持进度、保证质量、提高整个机构的生产效率。

在测试工作中应用软件缺陷管理系统具有以下优点：

- 保持高效率的测试过程。
- 提高软件缺陷报告的质量。
- 实施实时管理、安全控制。
- 利于项目组成员间的协同工作。

1. 缺陷跟踪系统常用的功能要求

缺陷跟踪系统常用的功能要求如下：

- 缺陷跟踪管理系统应采用 B/S 结构。
- 采用标准技术，基于网络、多功能、快速、可靠。
- 无需在用户终端机器上安装任何附加软件，可从网络上任何地方通过 HTTP 或 SMTP 连接。
- 支持网页界面，内部网和外部网的用户均可使用。
- 支持各种数据库系统，基于 SQL-92 标准，使用开放型数据库设计，可升级。
- 可设置性和多用途，不仅可用做 Bug 跟踪系统，而且可用做集成服务台热线流程管理。
- 支持各种自定域数据类型，表格栏目可根据需要添加和删减。

- 页面可做个性化调整，如添加公司标志等。
- 可同时支持多个项目、设置访问权限、自我注册。
- 可设置基于表格栏目域的多层次权限。
- 自定义工作流程（对每个项目）。
- 自动或手动分配任务负责人。
- 可一次附加多个任何类型的文件。
- 支持直接屏幕抓图（Screen Capture）。
- 具有自动自定义电子邮件通知的功能。
- 可通过电子邮件和文档版本管理软件集成。
- 图表报告、高级搜索、自定义常用搜索。
- 全面且易读的记录和修改历史。
- 快速排序和导出。
- 支持目录服务集成。
- 具有自定义电子邮件提醒催单功能。
- 方便的基于网络的系统管理。

2. 软件缺陷跟踪系统的选用

软件缺陷跟踪系统（缺陷管理工具）有英文版的，也有中文版的，并且收费和免费的均有。国内外已出现了一批质量较好的缺陷管理工具，其中比较有代表性的有：开源软件 Bugzilla、jira；Compuware 公司的 TrackRecord；Rational 公司的 ClearQuest 等。

选用软件缺陷跟踪系统时，需要注意该系统应具备如下要求：

- 安装简易、操作简易。
- 支持开发、构建、测试、验收多重迭代。
- 支持前台用户界面、后台缺陷数据库以及中间数据处理层。
- 显示测试工作的成效和项目的进展情况。
- 支持项目经理全程追踪督促。
- 支持开发组长、测试组长多级指派。
- 完整的追踪信息展现。
- 支持发现软件错误的类型、错误的严重等级。
- 支持发布版本的缺陷关联。
- Mail 实时通知缺陷任务。

16.4 软件测试的评估

软件测试的主要评测方法包括测试覆盖评估、质量评估、缺陷评估和性能评估。

16.4.1 测试覆盖评估

测试覆盖评估是对测试完全程度的评估，它建立在测试覆盖的基础上，测试覆盖是由测试需求、测试用例的覆盖或已执行代码的覆盖表示的。

覆盖指标提供了“测试的完全程度如何”这一问题的答案。最常用的测试覆盖评估是基于需求的测试覆盖和基于代码的测试覆盖。简而言之，测试覆盖是就需求（基于需求的）或代码的设计/实施标准（基于代码的）而言的完全程度的评估，如用例的核实（基于需求的）或所有代码行的执行（基于代码的）。

1. 基于需求的测试覆盖评估

基于需求的测试覆盖在测试生命周期中要评估多次，每一个测试阶段结束时给出测试覆盖的度量，并在测试生命周期的各阶段（里程碑）提供测试覆盖的标识（如基于需求的测试覆盖率、已计划的测试覆盖率、已实施的测试覆盖率、已执行的成功测试覆盖率）。

（1）基于需求的测试覆盖率

基于需求的测试覆盖率通过以下公式计算：

测试覆盖率=T(p,i,x,s)/RfT%

其中：T 是用测试过程或测试用例表示的测试（Test）数（已计划的、已实施的或成功的）。RfT 是测试需求（Requirement for Test）的总数。

（2）已计划的测试覆盖率

在制定测试计划活动中，已计划的测试覆盖率通过以下公式计算：

已计划的测试覆盖率=Tp/RfT %

其中：Tp 是用测试过程或测试用例表示的计划测试需求数，RfT 是测试需求的总数。

（3）已实施的测试覆盖率

在已实施的测试过程中，已实施的测试覆盖率可通过以下公式计算：

已实施的测试覆盖率=Ti/RfT %

其中：Ti 是用测试过程或测试用例表示的已执行的测试需求数。RfT 是测试需求的总数。

（4）已执行的成功测试覆盖率

在已执行的成功测试活动中，成功的测试覆盖率可通过以下公式计算：

成功的测试覆盖率=Ts/RfT %

其中：Ts 是用完全成功、没有缺陷的测试过程或测试用例表示的已执行测试需求数。RfT 是测试需求的总数。

2. 基于代码的测试覆盖评估

基于代码的测试覆盖评估是根据测试已经执行的源代码的多少来表示的，这种测试覆盖策略类型对于安全至上的系统来说非常重要。测试覆盖评估已经执行的代码的多少，与之相对的是要执行的剩余代码的多少。代码覆盖可以建立在控制流（语句、分支或路径）或数据流的基础上。控制流覆盖的目的是测试代码行、分支条件、代码中的路径或软件控制流的其他元素。数据流覆盖的目的是通过软件操作测试数据状态是否有效，例如，数据元素在使用之前是否已作定义。

基于代码的测试覆盖率可通过以下公式计算：

基于代码的测试覆盖率=Ie/Tiic %

其中：Ie 是用代码语句、代码分支、代码路径、数据状态判定点或数据元素名表示的已执行代码数，Tiic（Total number of Items in the code）是代码中的项目总数。

在软件测试评估工作中，基于代码的测试覆盖评估工作是很有意义的，因为任何未经测试的代码都是一个潜在的不利因素。在一般情况下，代码覆盖运用于较低的测试等级（例如单元和集成级）时最为有效，但是，仅仅凭借执行了所有的代码，并不能为软件质量提供保证，也就是说，即使所有的代码都在测试中得到执行，并不能说明代码是按照客户需求和设计的要求去做了。

16.4.2 软件测试的质量评估

质量是对测试对象（系统或测试的应用程序）的可靠性、稳定性以及性能的测评。质量建立在对测试结果的评估和对测试过程中确定的变更请求（缺陷）的分析的基础上。

对软件测试进行质量评估时需要重点注意如下内容。

1. 软件测试质量评估的准则

为了评估软件测试的质量，必须把不同特性的评价结果加以归纳，使用决策表或加权平均法，同时可以考虑其他因素，如在特定环境下对软件产品质量评估有影响的时间和成本。

软件测试的质量应根据质量特性的重要程度确定其权值，质量特性的重要程度可以根据软件类型的不同而有所不同。对于任务关键型的系统软件，其可靠性是最重要的；对于时间关键型的实时软件系统，其效率是最重要的；对于交互终端用户软件，其易用性是最重要的。

2. 软件测试质量评估的过程

软件测试质量评估的一般过程如下。

- 测量：把选定的度量应用到软件产品上去进行的活动。
- 评级：根据等级定义确定某一测量值的等级。
- 评估：将一组评出的等级加以归纳，得出软件产品质量报告。最后管理人员将归纳的质量与其他方面，诸如时间和成本进行比较，根据管理准则作出决策，决定该产品是否通过验收或者是否发行。

3. 软件测试质量评估的目的

软件测试质量评估的目的如下：

- 确定产品是否通过验收。
- 确定何时发布产品。
- 与其他类似产品相比较，对产品进行选择。
- 在使用该产品时评估其正面及负面的影响。
- 确定何时优化或替换该产品。

4. 软件测试质量评估的一般要求

软件测试质量评估的一般要求如下：

- 质量度量的选择。
- 对质量特性进行定义所采用的方式不提供对它们的直接测量，需要建立与软件产品的特性。
- 相关的度量。
- 度量可以因不同的环境和不同的开发阶段而异。
- 根据用户观点采用的度量是关键的。

5. 软件测试质量评估的等级

软件测试质量评估的等级需要注意如下 3 点内容：

- 对可定量的特征可用质量度量来定量地测量，将测试结果与预先定义好的等级（如规定达到什么程度为优秀，达到什么程度为良好，达到什么程度为合格，什么情况下为差）进行比较，得到该软件特性的评价结果。
- 质量与给定需求有关，不可能有通用的等级，每一次具体的评价都必须对等级进行定义。
- 应根据质量特性的重要程度确定权值。

6. 软件测试质量评估的内容

软件测试质量评估的主要内容如下。

- 功能性：功能性是指与一组功能及其指定的性质有关的属性。
- 可靠性：可靠性是指与在规定的一段时间和条件下，软件维持其性能水平的能力有关的一组属性。
- 易用性：易用性是指与一组规定或潜在的用户为使用软件所需作的努力和对这样的使用所作的评价有关的属性。
- 效率：效率是指与在规定的条件下，软件的性能水平与所使用资源量之间关系有关的一组属性。
- 可维护性：可维护性是指与进行指定的修改所需的努力有关的一组属性。
- 可移植性：可移植性是与软件可从某一环境转移到另一环境的能力有关的一组属性。

（1）功能性

功能性的主要内容如下。

- 适合性：与规定任务能否提供一组功能以及这组功能的适合程度有关的软件属性。
- 准确性：与能否得到正确、相符的结果或效果有关的软件属性。
- 互用性：与同其他指定系统进行交互能力有关的软件属性。
- 依从性：使软件遵循有关的标准、约定、法规及类似规定的软件属性。
- 安全性：与防止对程序及数据的非授权的故意或意外访问的能力有关的软件属性。

（2）可靠性

可靠性的主要内容如下。

- 成熟性：与由软件故障引起失效的频度有关的软件属性。
- 容错性：与在软件故障或违反指定接口的情况下，维持规定的性能水平的能力有关的软件

属性。

- 易恢复性：与在失效发生后，重建其性能水平并恢复直接受影响数据的能力，以及为达此目的所需的时间和能力有关的软件属性。

（3）易用性

易用性的主要内容如下。

- 易理解性：与用户为认识逻辑概念及其应用范围所花的努力有关的软件属性。
- 易学性：与用户为学习软件应用所花的努力有关的软件属性。
- 易操作性：与用户为操作和运行控制所花努力有关的软件属性。

（4）效率

与效率有关的内容如下。

- 时间特性：与软件执行其功能时响应、处理时间以及吞吐量有关的软件属性。
- 资源特性：与在软件执行其功能时所使用的资源数量及其使用时间有关的软件属性。

（5）可维护性

与可维护性有关的内容如下。

- 易分析性：与为诊断缺陷或失效原因及为判定待修改的部分所需努力有关的软件属性。
- 易改变性：与进行修改、排除错误或适应环境变化所需努力有关的软件属性。
- 稳定性：与修改所造成的未预料结果的风险有关的软件属性。
- 易测试性：与确认已修改软件所需的努力有关的软件属性。

（6）可移植性

与可移植性有关的内容如下。

- 适应性：与软件无需采用有别于为该软件准备的活动或手段就可能适应不同的规定环境有关的软件属性。
- 易安装性：与在指定环境下安装软件所需努力有关的软件属性。
- 遵循性：使软件遵循与可移植性有关的标准或约定的软件属性。
- 易替换性：与软件在该软件环境中用来替代指定的其他软件的机会和努力有关的软件属性。

7. 软件测试质量评估的产品描述

进行产品描述时需要注意如下事项。

- 性质：产品描述是产品软件包文档的一部分，提供关于用户文档、程序、数据的信息。
- 作用：帮助用户或潜在购买者作出产品是否适用的评价。
- 内容要求。

8. 软件测试质量评估的功能概述要求

软件测试质量评估的功能概述要求如下：

- 应概述产品的用户可调用的功能及其数据和设施。
- 边界值：应提供会致使产品的使用受限的特定边界值数据。
- 安全：应包含有关防止对程序和数据非授权访问的手段。
- 可靠性说明：应包含数据存储规程的信息；应描述保证产品的功能能力的附加性质；检验输入的合理性；防止由于用户的错误而产生的严重后果。
- 出错恢复。
- 易用性说明。
- 用户界面应命名用户界面的类型，如命令行、窗口、功能键要求的知识；应规定应用该产品所要求的专门知识和自然语言；适应用户的需要；应标识能被用户作适应性修改所需的工具和条件；防止侵权的行为；使用效率和用户满意度。
- 用户文档：用户文档应考虑完整性、正确性、一致性、易理解性、易浏览性，每个文档应有目录表、索引表，应提供打印方法。

9. 软件测试质量评估文档的检查测试

软件测试质量评估文档的检查测试应考虑如下要求：

- 测试细则。
- 细则概述。
- 测试预要求。
- 测试活动。
- 测试记录。
- 测试报告。
- 跟踪测试。

16.4.3 软件测试的缺陷评估

缺陷评估是对测试过程中缺陷达到的比率或发现的比率提供一个软件可靠性指标。对于缺陷分析，常用的主要缺陷参数有 4 个。

- 状态：缺陷的当前状态。
- 优先级：必须处理和解决缺陷的相对重要性。
- 严重性：最终用户、组织或第三方的影响等。
- 起源：导致缺陷的起源故障及其位置，或排除该缺陷需要修复的构件。

软件测试的缺陷评估可依据以下 4 类形式的度量提供缺陷测评：

- 缺陷发现率。
- 缺陷潜伏期。
- 缺陷分布（密度）。
- 整体软件缺陷清除率。

1. 缺陷发现率

缺陷发现率是将发现的缺陷数量作为时间的函数来评估，创建缺陷趋势图或报告，如图 16-3

所示。

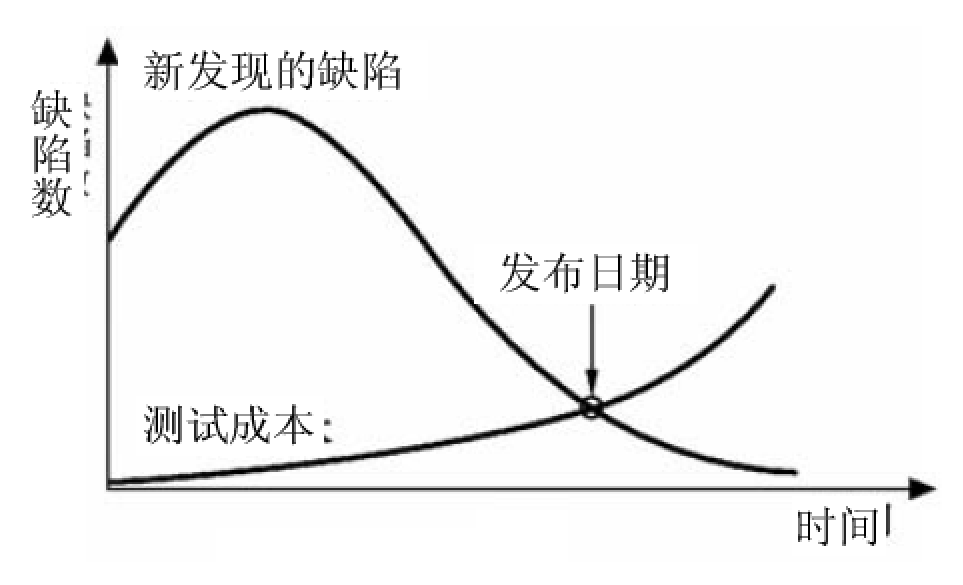


图 16-3　缺陷发现率

对图 16-3 的说明如下：

- 缺陷发现率将随着测试时间和修复进度而减少。
- 缺陷发现率将随着测试时间而测试成本增加。
- 可以设定一个阈值，在缺陷发现率低于该阈值时才能部署软件。

图 16-3 描述的是已报告的缺陷，然而，需要查看并分析一下，为什么许多报告的缺陷不是重复的缺陷就是未经确认的缺陷，这样做是很有价值的。

2. 缺陷潜伏期

Rational Unified Process 提供缺陷潜伏期（也称为阶段潜伏期、缺陷龄）评估，缺陷潜伏期报告是一种特殊类型的缺陷分布报告。缺陷潜伏期报告显示缺陷处于特定状态下的时间长短。缺陷潜伏期是一种特殊类型的缺陷分布度量。在实际测试工作中，发现缺陷的时间越晚，这个缺陷所带来的损害就越大，修复这个缺陷所耗费的成本就越多。表 16-7 显示了一个项目的缺陷潜伏期的度量。

表 16-7　一个项目的缺陷潜伏期的度量

缺陷造成阶段	发现阶段									
	需求	总体设计	详细设计	编码	单元测试	集成测试	系统测试	验收测试	试运行产品	发布产品
需求	0	1	2	3	4	5	6	7	8	9
总体设计		0	1	2	3	4	5	6	7	8
详细设计			0	1	2	3	4	5	6	7
编码				0	1	2	3	4	5	6
总计										

在上表中，总体设计发现缺陷，阶段潜伏期为 1，在发布产品时，阶段潜伏期为 9。

3. 缺陷分布

缺陷分布（密度）报告允许将缺陷计数作为一个或多个缺陷参数的函数来显示。软件缺陷分布（密度）是一种以平均值估算法来计算出软件缺陷分布（密度）值。程序代码通常是以千行为单位的，软件缺陷分布（密度）是用下面的公式计算的：

$$软件缺陷密度=\frac{软件缺陷数量}{代码行或功能点的数量}$$

4. 整体软件缺陷清除率

为了估算软件缺陷清除率，首先需要引入几个变量，F 为描述软件规模用的功能点，D1 为软件开发过程中发现的所有软件缺陷数，D2 为软件分布后发现的软件缺陷数，D 为发现的总软件缺陷数。由此可得到 D=D1+D2 的关系。

对于一个软件项目，可用如下几个公式从不同角度来估算软件的质量：

- 质量（每个功能点的缺陷数）=D2/F。
- 软件缺陷注入率=D/F。
- 整体软件缺陷清除率=D1/D。

16.4.4　软件测试的性能评估

评估测试对象的性能行为时，可以使用多种评测，这些评测侧重于获取与行为相关的数据，如响应时间、计时配置文件、执行流、操作可靠性和限制。这些评测主要在“评估测试”活动中进行评估，但是也可以在“执行测试”活动中使用性能评测评估测试进度和状态，主要的性能评测包括以下几点。

- 动态监测：在测试执行过程中，实时获取并显示正在执行的各测试脚本的状态。
- 响应时间和吞吐量：测试对象针对特定主角、用例的响应时间或吞吐量的评测。
- 百分比报告：数据已收集值的百分位评测/计算。
- 比较报告：代表不同测试执行情况的两个（或多个）数据集之间的差异或趋势。
- 追踪和配置文件报告：测试用例和测试对象之间的消息和会话详细信息。

1. 动态监测

动态监测通常是以柱状图或曲线图的形式提供实时显示/报告。该报告用于在测试执行过程中，通过显示当前的情况、状态以及测试用例正在执行的进度来监测或评估性能测试的执行情况，如图 16-4 所示。

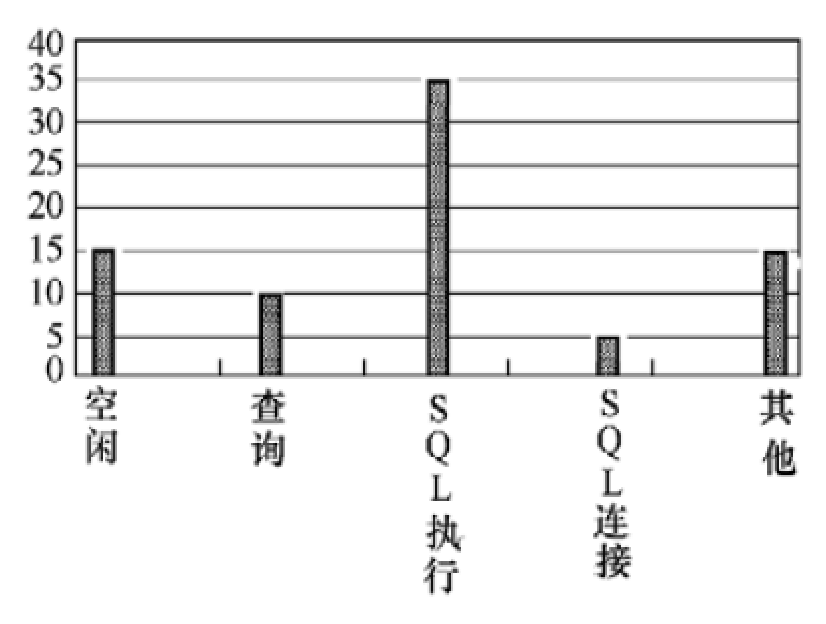


图 16-4　动态监测柱状图

图 16-4 中共有 80 个测试用例在执行，如图 16-5 所示。

空闲： 15个
查询： 10个
SQL执行： 35个
SQL 连接： 5个
其他： 15个

图 16-5 测试用例

2. 响应时间和吞吐量

响应时间和吞吐量是评测并计算与时间和吞吐量相关的性能行为。这些报告通常用曲线图显示，如图 16-6 所示。

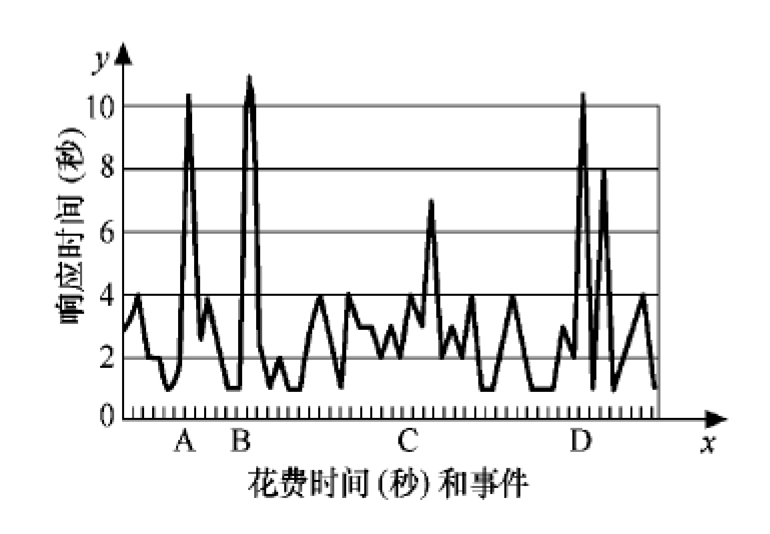


图 16-6 响应时间和吞吐量

3. 百分比报告

百分比报告通过显示已收集数据类型的各种百分比值，提供了另一种性能统计计算的方法，如图 16-7 所示。

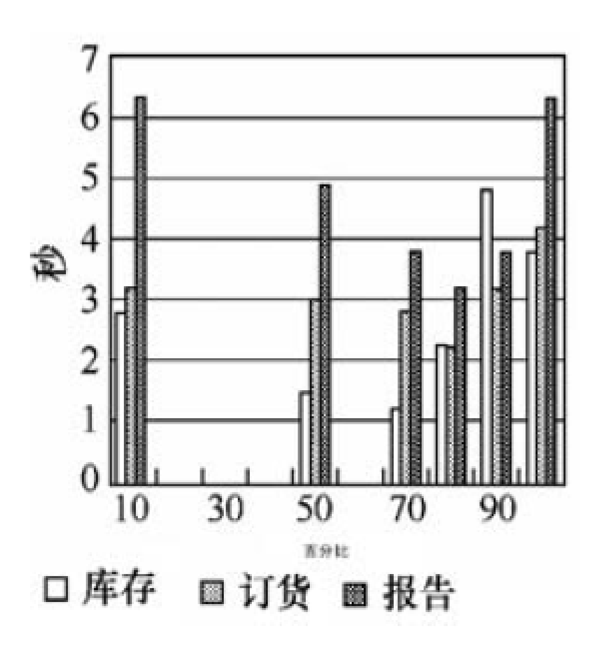


图 16-7 百分比报告

4. 比较报告

比较不同性能测试的结果，以评估测试执行过程中所作的变更对性能行为的影响，这种做法是非常必要的。比较报告应该用于显示两个数据集（分别代表不同的测试执行）之间的差异或多个测试执行之间的趋势。

5. 追踪和配置文件报告

当性能行为可以接受时，或性能监测表明存在可能的瓶颈时（如当测试用例保持给定状态的时间过长），追踪报告可能是最有价值的报告。追踪和配置文件报告显示低级信息，该信息包括主角与测试对象之间的消息、执行流、数据访问以及函数和系统调用等。

第17章　测试用例设计和电子政务应用平台测试用例设计实例

测试用例反映对被测对象的质量要求和评估范围，决定测试的效率和测试自身的质量。测试用例的设计是整个软件测试工作的核心，设计人员要以一些比较成熟的测试用例设计方法为指导，结合个人的经验积累来设计测试用例。

本章重点讨论以下内容：

- 测试用例的基本概念。
- 界面测试用例设计实例。
- 登录、添加、删除、查询模块测试用例设计实例。
- 宽带接入网网络管理系统测试用例设计实例。
- 某部电子政务应用平台测试用例设计实例。
- 电子政务应用平台主页功能测试用例设计实例。

17.1　测试用例的基本概念

17.1.1　测试用例概述

1. 测试用例的定义

测试用例是一个文档，是执行的最小实体。测试用例描述输入、动作、时间和一个期望的结果，其目的是确定应用程序的某个特性是否正常的工作，并且达到程序所设计的结果，以便测试某个程序路径或核实是否满足某个特定需求。

测试用例目前没有经典的定义，比较通常的说法是指对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术和策略；内容包括测试目标、测试环境、输入数据、测试步骤、预期结果、测试脚本等，并形成文档。

测试用例是软件测试的核心，不同类别的软件测试的用例是不同的。

2. 测试用例的优点

测试用例具有如下11个优点：

- 在开始实施测试之前设计好测试用例，避免盲目测试并提高测试效率，减少测试的不完全性。
- 测试用例使软件测试的实施重点突出、目的明确。

- 根据测试用例的多少和执行难度，估算测试工作量，便于测试项目的时间和资源管理与跟踪。
- 减少回归测试的复杂程度。
- 在软件版本更新后只需修正少量的测试用例便可展开测试工作、降低工作强度、缩短项目周期。
- 功能模块的测试用例的通用化和复用化会使软件测试易于开展。
- 根据测试用例的操作步骤和执行结果，可以方便地书写软件测试缺陷报告。
- 可以根据测试用例的执行等级，实施不同级别的测试。
- 为分析软件缺陷和程序模块质量提供依据。
- 可以最大程度地找出软件隐藏的缺陷。
- 测试用例内容清晰、格式一致、分类组织。

3. 测试用例的选择原则

测试用例的选择原则如下：

- 如果输入条件规定了值的范围，则应取刚达到这个范围的边界值以及刚刚超过这个范围边界的值作为测试输入数据。
- 如果输入条件规定了值的个数，则用最大个数、最小个数和比最大个数多 1 个、比最小个数少 1 个的数作为测试数据。
- 根据程序规格说明的每个输出条件。
- 如果程序的规格说明给出的输入域或输出域是有序集合（如有序表、顺序文件等），则应选取集合中的第一个和最后一个元素作为测试用例。
- 如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构的边界上的值作为测试用例。
- 分析程序规格说明，找出其他可能的边界条件。

4. 测试用例的 4 性

测试用例的 4 性是指代表性、针对性、可判定性、可重现性。

- 测试用例的代表性：能够代表并覆盖各种合理的和不合理的、合法的和非法的、边界的和越界的以及极限的输入数据、操作和环境设置等。
- 测试用例的针对性：对程序中可能存在的错误有针对性地测试软件缺陷。
- 测试结果的可判定性：测试执行结果的正确性是可判定的，每一个测试用例都应有相应的期望结果。
- 测试结果的可重现性：对同样的测试用例，系统的执行结果应当是相同的。

5. 测试用例的组成元素

测试用例通常包括以下几个组成元素：

- 用例 ID。
- 用例名称。
- 测试目的。

- 测试级别。
- 参考信息。
- 测试环境。
- 前提条件。
- 测试步骤。
- 预期结果。
- 设计人员。

6. 测试用例在软件测试中的作用

（1）指导测试的实施

在实施测试时测试用例作为测试的标准，测试人员一定要按照测试用例中的用例项目和测试步骤逐一实施测试，并将测试情况记录在测试用例管理软件中，以便自动生成测试结果文档。

（2）根据测试的要求设计测试用例

单元测试应测试的用例、集成测试应测试的用例、系统测试和回归测试应测试的用例，在设计测试用例时都已作明确规定，实施测试时测试人员不能随意进行变动。

（3）提供评估测试结果的度量基准

完成测试实施后需要对测试结果进行评估，并且编制测试报告。测试用例提供判断软件测试是否完成、衡量测试质量需要一些量化的结果。

7. 测试用例的分类

测试用例是分类的，要是没有分类的用例则将不便于维护和阅读。

测试用例可分为如下几类：

- 接口测试用例。
- 路径测试用例。
- 功能测试用例。
- 容错能力测试用例。
- 性能测试用例。
- 用户界面测试用例。
- 信息安全测试。
- 压力测试用例。
- 可靠性测试用例。
- 安装/反安装测试用例。

17.1.2 测试用例设计

对于一个测试人员来说，测试用例的设计编写是一项必须掌握的能力，但有效的设计和熟练的编写测试用例却是一个十分复杂的技术，测试用例编写者不仅要掌握软件测试的技术和流程，而且还要对整个软件不管从业务上，还是对被测软件的设计、功能规格说明、用户试用场景以及程序

/模块的结构等方面，都有比较透彻地理解和明晰地把握，稍有不慎就会顾此失彼，造成疏漏。

测试用例的设计方法不是单独存在的，具体到每个测试项目里都会用到多种方法，每种类型的软件都有各自的特点，每种测试用例设计的方法也有各自的特点，针对不同软件有不同的设计方法。

进行测试用例设计时需要考虑的因素如下。

1. 设计测试用例时依据的文档和资料

编写测试用例所依据的文档和资料如下：

- 软件需求说明及相关文档。
- 相关的设计说明（概要设计、软件需求文档、软件设计文档、详细设计等）。
- 与开发组交流对需求理解的记录。
- 已经基本成熟的测试用例。

2. 设计测试用例时的基本原则

设计测试用例时的基本原则如下：

- 利用成熟的测试用例设计方法来指导设计。
- 测试用例的正确性。
- 测试用例的代表性。
- 测试结果的可判定性。
- 测试结果的可重现性。
- 足够详细、准确和清晰的步骤。
- 利用测试用例文档编写测试用例时必须符合内部的规范要求。

3. 设计测试用例时应注意的问题

设计测试用例时应注意如下问题：

- 不能把测试用例设计等同于测试输入数据的设计。
- 不能追求测试用例设计的一步到位。
- 不能将多个测试用例混在一个用例中。
- 不能由没有测试经验的人员设计测试用例。
- 测试用例文档由简介和测试用例两部分组成：简介部分编制了测试目的、测试范围、定义术语、参考文档、概述等；测试用例部分逐一列出各测试用例，每个具体测试用例都将包括下列详细信息：用例编号、用例名称、测试等级、入口准则、验证步骤、期望结果（含判断标准）、出口准则、注释等。

测试用例的编写如表 17-1 所示。

表 17-1　测试用例的编写表

<table>
<tr><td colspan="2">测试标题</td><td colspan="3"></td><td>用例的编号 ID</td><td colspan="3"></td></tr>
<tr><td colspan="2">测试技术</td><td colspan="2"></td><td>测试环境要求</td><td></td><td colspan="2">特殊要求</td><td></td></tr>
<tr><td colspan="2">测试用例设计人员</td><td colspan="2"></td><td>测试人员</td><td></td><td colspan="2">测试日期</td><td></td></tr>
<tr><td colspan="2">测试目的</td><td colspan="7"></td></tr>
<tr><td colspan="2">测试对象</td><td colspan="7"></td></tr>
<tr><td>测试项</td><td colspan="2">测试内容</td><td colspan="2">测试方法与步骤</td><td colspan="2">测试判断准则</td><td colspan="2">测试结果</td></tr>
<tr><td>1</td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td>2</td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td>3</td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td>⋮</td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td>n</td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td></tr>
</table>

进行测试用例的评审检查时应注意如下内容：

- 《需求规格说明书》是否评审并建立了基线?
- 软件需求规格说明中规定是否评审?
- 是否按照测试计划时间完成用例编写?
- 需求新增和变更是否进行了对应的调整?
- 用例是否按照模板进行编写?
- 测试用例是否覆盖了《需求规格说明书》?
- 计划的测试用例的个数。
- 测试用例设计是否全面准确?
- 是否针对不同的数据类型?
- 用例编号是否和需求进行对应?
- 非功能测试需求或不可测试需求是否在用例中列出并说明?
- 用例设计是否包含了正面、反面的用例?
- 每个测试用例是否清楚地填写了测试特性、步骤、预期结果?
- 步骤/输入数据部分是否清晰，是否具备可操作性?
- 测试用例是否包含测试数据、测试数据的生成办法或者输入的相关描述?
- 是否针对需求的不同部分设计使用不同的设计方法?
- 每个测试用例是否都阐述预期结果和评估该结果的方法?
- 需要进行打印表格，导入、导出、接口是否存在打印位置、表格名称，指定数据库表名或文件位置，表格和数据格式是否有说明或附件?
- 用例覆盖率是否达到相应质量指标?
- 用例预期缺陷率是否达到相应质量指标?

17.2　界面测试用例设计实例

界面是软件与用户交互的最直接的层，界面的好坏决定着用户对软件的第一印象。设计合理的界面能给用户带来轻松愉悦的感受和成功的感觉，相反，由于界面设计不好，将使用户反感。

目前界面风格主要有三种方式：多窗体、单窗体以及资源管理器方式，无论哪种方式，界面都要进行测试，界面测试用例设计实例如下。

- 测试目的：检验界面的可用性、检验界面的可靠性、检验用户界面是否友好。
- 测试对象：测试的对象为界面。
- 测试内容：易用性、规范性、帮助设施、合理性、美观与协调性、菜单位置、独特性、快捷方式的组合、安全性、多窗口的应用与系统资源、文本框、命令按钮控件、单选按钮控件、up-down 控件文本框、组合列表框、复选框、列表框控件、滚动条控件、各种控件在窗体中混合使用、插入操作、编辑操作、窗体、控件、菜单。

1. 易用性

易用性是指按钮名称应该易懂，用词准确，要与同一界面上的其他按钮易于区分，能望文知意。

对易用性的测试说明如表 17-2 所示。

表 17-2　对易用性的测试说明

测试内容	测试方法与步骤	测试判断准则
常用按钮要支持快捷方式	完成相同或相近功能的按钮使用 Frame 框起来	是否准确
减少鼠标移动的距离	完成同一功能或任务的元素放在集中位置	是否准确
常用组合快捷键 Ctrl+Tab	分页界面要支持在页面间的快捷切换	是否准确
默认按钮要支持 Enter 操作	按 Enter 后自动执行默认按钮的对应操作	是否准确
复选框和选项框要有默认选项	复选框和选项框要有默认选项，并支持 Tab 选择	是否准确
选项数相同时多用选项框而不用下拉列表框	操作	是否准确
界面空间较小时使用下拉列表框，相反使用选项框	操作	是否准确
选项数较少时使用选项框，相反使用下拉列表框	操作	是否准确

2. 规范性

通常界面设计都按 Windows 界面的规范来设计，即包含菜单栏、工具栏、状态栏、滚动条、右键快捷菜单等的标准格式。

对规范性的测试说明如表 17-3 所示。

表 17-3　对规范性的测试说明

测试内容	测试方法与步骤	测试判断准则
常用菜单要有命令快捷方式	观看	是否符合规范
完成相同或相近功能的菜单用横线隔开放在同一位置	观看	是否符合规范
菜单前的图标能直观地代表要完成的操作	观看	是否符合规范
菜单深度一般要求最多控制在三层以内	观看	是否符合规范
工具栏要求可以根据用户的要求自己选择定制	观看	是否符合规范
相同或相近功能的工具栏放在一起	观看	是否符合规范
工具栏中的每一个按钮要有及时提示信息	观看	是否符合规范

（续表）

测试内容	测试方法与步骤	测试判断准则
工具栏的长度最长不能超出屏幕宽度	观看	是否符合规范
工具栏的图标能直观地代表要完成的操作	观看	是否符合规范
系统常用的工具栏设置的默认放置位置	观看	是否符合规范
工具栏太多时可以考虑使用工具厢	观看	是否符合规范
工具厢要具有可增减性，由用户自己根据需求定制	观看	是否符合规范
工具厢的默认总宽度不要超过屏幕宽度的 1/5	观看	是否符合规范
状态条要能显示用户切实需要的信息，常用的有：目前的操作、系统状态、用户位置、用户信息、提示信息、错误信息等，如果某一操作需要的时间较长，还应该显示进度条和进程提示	观看	是否符合规范
滚动条的长度要根据显示信息的长度或宽度及时变换，以利于用户了解显示信息的位置和百分比	观看	是否符合规范
状态条的高度以放置好字为宜，滚动条的宽度要比状态条略窄	观看	是否符合规范
菜单和工具条要有清楚的界限，菜单要求凸出显示，这样在移走工具条时仍有立体感	观看	是否符合规范
菜单和状态条中通常使用 5 号字体。工具条一般比菜单要宽，但不要宽的太多，否则看起来很不协调	观看	是否符合规范
右键快捷菜单采用与菜单相同的准则	观看	是否符合规范

3. 帮助设施

系统应该提供详尽而可靠的帮助文档，在用户使用产生困惑时可以自己寻求解决方法。

对帮助设施的测试说明如表 17-4 所示。

表 17-4　对帮助设施的测试说明

测试内容	测试方法与步骤	测试判断准则
帮助文档中的性能介绍与说明要与系统性能配套一致	观看	是否符合帮助
打包新系统时，对作了修改的地方在帮助文档中要做相应的修改	观看	是否符合帮助
操作时要提供及时调用系统帮助的功能	观看	是否符合帮助
在界面上调用帮助时应该能够及时定位到与该操作相对的帮助位置，也就是说帮助要有即时针对性	观看	是否符合帮助
提供目前流行的联机帮助格式或 HTML 帮助格式	观看	是否符合帮助
用户可以用关键词在帮助索引中搜索所要的帮助，当然也应该提供帮助主题词	观看	是否符合帮助
如果没有提供书面帮助文档的话，最好有打印帮助的功能	观看	是否符合帮助
在帮助中应该提供技术支持的方式，一旦用户难以自己解决可以方便地寻求新的帮助方式	观看	是否符合帮助

4. 合理性

屏幕对角线相交的位置是用户直视的地方，正上方四分之一处为易吸引用户注意力的位置，在放置窗体时要注意利用这两个位置。

对合理性的测试说明如表 17-5 所示。

表 17-5　对合理性的测试说明

测试内容	测试方法与步骤	测试判断准则
父窗体或主窗体的中心位置应该在对角线的焦点附近	观看	是否合理
子窗体位置应该在主窗体的左上角或正中	观看	是否合理
多个子窗体弹出时应该依次向右下方偏移，以显示窗体的标题为宜	观看	是否合理
重要的命令按钮与使用较频繁的按钮要放在界面上注目的位置	观看	是否合理
错误使用容易引起界面退出，关闭的按钮不应该放在易点位置。横排开头/最后与竖排最后为易点位置	观看	是否合理
与正在进行的操作无关的按钮应该加以屏蔽	观看	是否合理
对可能造成数据无法恢复的操作必须提供确认信息，给用户放弃选择的机会	观看	是否合理
非法的输入或操作应有足够的提示说明	观看	是否合理
对运行过程中由出现问题而引起错误的地方要有提示，让用户明白错误出处，避免形成无限期地等待	观看	是否合理
提示、警告或错误说明应该清楚、明了、恰当	观看	是否合理

5. 美观与协调性

界面应该大小适合美学观点，感觉协调舒适，能在有效范围内吸引用户的注意力。

对美观与协调性的测试说明如表 17-6 所示。

表 17-6　对美观和协调性的测试说明

测试内容	测试方法与步骤	测试判断准则
长宽接近于黄金点比例，切忌长宽比例失调或宽度超过长度	观看、操作	是否美观、协调
布局要合理，不宜过于密集，也不能过于空旷，合理的利用空间	观看、操作	是否美观、协调
按钮大小基本相近，忌用太长的名称，免得占用过多的界面位置	观看、操作	是否美观、协调
按钮的大小要与界面的大小和空间协调	观看、操作	是否美观、协调
避免空旷的界面上放置很大的按钮	观看、操作	是否美观、协调
放置控件后界面不应有很大的空缺位置	观看、操作	是否美观、协调
字体的大小要与界面的大小比例协调	观看、操作	是否美观、协调
前景与背景色搭配合理协调，反差不宜太大，最好少用深色，如大红、大绿等。常用色考虑使用 Windows 界面色调	观看、操作	是否美观、协调
如果使用其他颜色，主色要柔和，具有亲和力与磁力，杜绝刺目的颜色	观看、操作	是否美观、协调
界面风格要保持一致，字的大小、颜色、字体要相同，除非是需要艺术处理或有特殊要求的地方	观看、操作	是否美观、协调
如果窗体支持最小化、最大化或放大时，窗体上的控件也要随着窗体而缩放；切忌只放大窗体而忽略控件的缩放	观看、操作	是否美观、协调
对于含有按钮的界面一般不应该支持缩放，即右上角只有关闭功能	观看、操作	是否美观、协调
通常父窗体支持缩放时，子窗体没有必要缩放	观看、操作	是否美观、协调
如果能给用户提供自定义界面风格则更好，由用户自己选择颜色、字体等	观看、操作	是否美观、协调

6. 菜单位置

菜单是界面上最重要的元素，菜单位置应按照功能来组织。

对菜单的测试说明如表 17-7 所示。

表 17-7　对菜单的测试说明

测试内容	测试方法与步骤	测试判断准则
菜单通常采用“常用”、“主要”、“次要”、“工具”、“帮助”的位置排列，符合流行的 Windows 风格	观看、操作	是否美观、协调
常用的有“文件”、“编辑”、“查看”等，几乎每个系统都有这些选项，当然要根据不同的系统有所取舍	观看、操作	是否美观、协调
下拉菜单要根据菜单选项的含义进行分组，并按照一定的规则进行排列，用横线隔开	观看、操作	是否美观、协调
一组菜单的使用有先后要求或有向导作用时，应该按先后次序排列	观看、操作	是否美观、协调
没有顺序要求的菜单项按使用频率和重要性排列，常用的放在开头，不常用的靠后放置；重要的放在开头，次要的放在后边	观看、操作	是否美观、协调
如果菜单选项较多，应该采用加长菜单的长度而减少深度的原则排列	观看、操作	是否美观、协调
菜单深度一般要求最多控制在三层以内	观看、操作	是否美观、协调
对常用的菜单要有快捷命令方式	观看、操作	是否美观、协调
对与进行的操作无关的菜单要用屏蔽的方式加以处理	观看、操作	是否美观、协调
菜单前的图标不宜太大，与字高保持一致	观看、操作	是否美观、协调
主菜单的宽度要接近，字数不应多于 4 个	观看、操作	是否美观、协调
主菜单数目不应太多，最好为单排布置	观看、操作	是否美观、协调
如果能给用户提供自定义界面风格则更好，由用户自己选择颜色、字体等	观看、操作	是否美观、协调

7. 独特性

如果一味遵循业界的界面标准，则会丧失自己的个性。在框架符合以上规范的情况下，设计具有自己独特风格的界面尤为重要，尤其是在商业软件流通中有着很好的迁移默化的广告效果。

对独特性的测试说明如表 17-8 所示。

表 17-8　对独特性的测试说明

测试内容	测试方法与步骤	测试判断准则
安装界面上应有单位介绍或产品介绍，并有自己的图标	观看、操作	是否美观、协调
主界面，最好是大多数界面上要有公司图标	观看、操作	是否美观、协调
登录界面上要有本产品的标志，同时包含公司图标	观看、操作	是否美观、协调
帮助菜单的“关于”中应有版权和产品信息	观看、操作	是否美观、协调
公司的系列产品要保持一致的界面风格，如背景色、字体、菜单排列方式、图标、安装过程、按钮用语等应该大体一致	观看、操作	是否美观、协调
安装界面上应有单位介绍或产品介绍，并有自己的图标	观看、操作	是否美观、协调

8. 快捷方式的组合

在菜单及按钮中使用快捷键可以让喜欢使用键盘的用户操作得更快，在 Windows 及其应用软件中快捷键的使用方法大多是一致的。

对快捷方式的测试说明如表 17-9 所示。

表 17-9 对快捷方式的测试说明

测试内容	测试方法与步骤	测试判断准则
面向事务的组合，如 Ctrl+D 用于设置字体；Ctrl+F 用于查找；Ctrl+H 用于替换；Ctrl+I 用于插入斜体；Ctrl+N 用于新建；Ctrl+S 用于 保存；Ctrl+O 用于打开	观看、操作	是否组合
列表，如 Ctrl+G 用于定位	观看、操作	是否组合
编辑，如 Ctrl+A 用于全选；Ctrl+C 用于复制；Ctrl+V 用于粘贴；Ctrl+X 用于剪切；Ctrl+Z 用于撤消操作；Ctrl+Y 用于恢复操作	观看、操作	是否组合
文件操作，如 Ctrl+P 用于打印；Ctrl+W 用于关闭	观看、操作	是否组合
系统菜单，如 Alt+A 用于弹出“表格”菜单；Alt+E 用于“编辑”菜单；Alt+T 用于“工具”菜单；Alt+W 用于“窗口”菜单；Alt+H 用于“帮助”菜单	观看、操作	是否组合
MS Windows 保留键，如 Ctrl+Esc 用于打开“开始”菜单；Ctrl+F4 用于关闭窗口；Alt+F4 用于结束应用；Alt+Tab 用于下一应用；Enter 用于默认按钮/确认操作；Esc 用于取消按钮/取消操作；Shift+F1 用于显示格式	观看、操作	是否组合
按钮的快捷键可以根据系统需要而调节，以下只是常用的组合，如 Alt+Y 用于确定（是）；Alt+C 用于取消；Alt+N 用于否；Alt+D 用于删除；Alt+Q 用于退出；Alt+A 用于添加；Alt+E 用于编辑；Alt+B 用于浏览；Alt+R 用于读；Alt+W 用于写	观看、操作	是否组合

9. 安全性考虑

在界面上控制出错几率，会大大减少系统因用户人为的错误引起的破坏。开发者应当尽量周全地考虑到各种可能发生的问题,使出错的可能降至最小。

对安全性考虑的测试说明如表 17-10 所示。

表 17-10 对安全性考虑的测试说明

测试内容	测试方法与步骤	测试判断准则
排除可能会使应用非正常中止的错误	观看、操作	是否安全
应当注意尽可能避免用户无意录入无效的数据	观看、操作	是否安全
采用相关控件限制用户输入值的种类	观看、操作	是否安全
当用户作出选择的可能性只有两个时，可以采用单选框	观看、操作	是否安全
当选择的可能再多一些时，可以采用复选框，每一种选择都是有效的，用户不可能输入任何一种无效的选择	观看、操作	是否安全
当选项特别多时，可以采用列表框、下拉式列表框	观看、操作	是否安全
在一个应用系统中，开发者应当避免用户作出未经授权或没有意义的操作	观看、操作	是否安全
对可能引起致命错误或系统出错的输入字符或动作要加以限制或屏蔽	观看、操作	是否安全

（续表）

测试内容	测试方法与步骤	测试判断准则
对可能发生严重后果的操作要有补救措施。通过补救措施用户可以回到原来的正确状态	观看、操作	是否安全
对一些特殊符号的输入、与系统使用的符号相冲突的字符等进行判断并阻止用户输入该字符	观看、操作	是否安全
对错误操作最好支持可逆性处理，如取消系列操作	观看、操作	是否安全
在输入有效性字符之前应该阻止用户进行只有输入之后才可进行的操作	观看、操作	是否安全
对可能造成等待时间较长的操作应该提供取消功能	观看、操作	是否安全
与系统采用的保留字符冲突的要加以限制	观看、操作	是否安全
在读入用户所输入的信息时，根据需要选择是否去掉前后空格	观看、操作	是否安全
有些读入数据库的字段不支持中间有空格，但用户切实需要输入中间空格，这时要在程序中加以处理	观看、操作	是否安全

10. 多窗口的应用与系统资源

设计良好的软件不仅要具有完备的功能，而且要尽可能地占用最低限度的资源。

对多窗口的应用与系统资源的测试说明如表 17-11 所示。

表 17-11 对多窗口的应用与系统资源的测试说明

测试内容	测试方法与步骤	测试判断准则
在多窗口系统中，有些界面要求必须保持在最顶层，避免用户在打开多个窗口时，不停地切换甚至最小化其他窗口来显示该窗口	观看、操作	是否多窗口应用
在主界面载入完毕后自动退出内存，让出所占用的 Windows 系统资源	观看、操作	是否多窗口应用
关闭所有窗体，系统退出后要释放所占的所有系统资源，除非是需要后台运行的系统	观看、操作	是否多窗口应用
尽量防止对系统的独占使用	观看、操作	是否多窗口应用

11. 文本框

对文本框的测试说明如表 17-12 所示。

表 17-12 对文本框的测试说明

测试内容	测试方法与步骤	测试判断准则
输入正常的字母或数字	观看、操作	出错提示
输入已存在的文件名称	观看、操作	出错提示
输入超长字符，如在“名称”文本框中输入超过允许边界个数的字符，假设最多 255 个字符，尝试输入 256 个字符，检查程序能否正确处理	观看、操作	出错提示
输入默认值、空白、空格	观看、操作	出错提示
若只允许输入字母，则尝试输入数字；反之；尝试输入字母	观看、操作	出错提示
利用复制、粘贴等操作强制输入程序不允许的输入数据	观看、操作	出错提示

（续表）

测试内容	测试方法与步骤	测试判断准则
输入特殊字符集，如 NULL 及\n 等	观看、操作	出错提示
输入超过文本框长度的字符或文本，检查所输入的内容是否正常显示	观看、操作	出错提示
输入不符合格式的数据，检查程序是否正常校验，如程序要求输入年月日格式为 yy/mm/dd，实际输入 yyyy/mm/dd，程序应该给出错误提示	观看、操作	出错提示
输入非法数据	观看、操作	出错提示
输入默认值	观看、操作	出错提示
输入特殊字符集	观看、操作	出错提示
输入使缓冲区溢出的数据	观看、操作	出错提示
输入相同的文件名	观看、操作	出错提示

12. 命令按钮控件

对命令按钮控件的测试说明如表 17-13 所示。

表 17-13　对命令按钮控件的测试说明

测试内容	测试方法与步骤	测试判断准则
单击按钮正确响应操作	单击“确定”按钮，是否正确执行操作；单击“取消”按钮，是否退出窗口	正确响应操作
非法的输入或操作	对非法的输入或操作给出足够的提示说明，如输入月工作天数为 32 时，单击“确定”按钮后系统应提示：天数不能大于 31	提示说明
对可能造成数据无法恢复的操作	对可能造成数据无法恢复的操作必须给出确认信息，给用户放弃选择的机会	必须给出确认信息

13. 单选按钮控件

对单选按钮控件的测试说明如表 17-14 所示。

表 17-14　对单选按钮控件的测试说明

测试内容	测试方法与步骤	测试判断准则
单选按钮	一组按钮中选择按钮	只能选中一个
逐一执行每个单选按钮	一组按钮中选择按钮	只能选中一个
一组执行同一功能的单选按钮	一组执行同一功能的单选按钮，在初始状态时必须有一个被默认选中	不能同时为空

14. up-down 控件文本框

对 up-down 控件文本框的测试说明如表 17-15 所示。

表 17-15　对 up-down 控件文本框的测试说明

测试内容	测试方法与步骤	测试判断准则
直接输入数字或用上下箭头控制	在“数目”文本框中直接输入 10，或者单击向上的箭头，使数目变为 10	

（续表）

测试内容	测试方法与步骤	测试判断准则
利用上下箭头控制数字的自动循环	当最多数字为 253 时，单击向上箭头，数目自动变为 1；反之亦适用	
直接输入超边界值	直接输入超边界值	系统应该提示重新输入
输入默认值，空白	“插入”为默认值，单击“确定”按钮；或删除默认值，使内容为空，单击“确定”按钮进行测试	应提示输入有误
输入字符	输入字符	

15. 组合列表框

对组合列表框的测试说明如表 17-16 所示。

表 17-16　对组合列表框的测试说明

测试内容	测试方法与步骤	测试判断准则
详细条目内容	详细条目内容可以根据需求说明确定	内容正确
逐一执行列表框中每个条目的功能	观看、操作	正确
检查能否向组合列表框中输入数据	观看、操作	正确

16. 复选框

对复选框的测试说明如表 17-17 所示。

表 17-17　对复选框的测试说明

测试内容	测试方法与步骤	测试判断准则
多个复选框可以被同时选中	观看、操作	不能
多个复选框可以被部分选中	观看、操作	不能
多个复选框可以都不被选中	观看、操作	不能
逐一执行每个复选框的功能	观看、操作	能

17. 列表框控件

对列表框控件的测试说明如表 17-18 所示。

表 17-18　对列表框控件的测试说明

测试内容	测试方法与步骤	测试判断准则
详细条目内容	根据需求说明书确定列表的各项内容	内容正确
使用滚动条	观看、操作	内容正确
列表框多选	分别检查利用 Shift 选中条目、利用 Ctrl 选中条目和直接利用鼠标选中多项条目的情况	内容正确

18. 滚动条控件

对滚动条控件的测试说明如表 17-19 所示。

表 17-19　对滚动条控件的测试说明

测试内容	测试方法与步骤	测试判断准则
滚动条	观看、操作	滚动条的长度根据显示信息的长度或宽度及时变换，这样有利于用户了解显示信息的位置和百分比
拖动滚动条	观看、操作	检查屏幕刷新情况，并查看是否有乱码
单击滚动条	观看、操作	正确
用滚轮控制滚动条	观看、操作	正确
滚动条的上下按钮	观看、操作	正确

19. 各种控件在窗体中混合使用

对各种控件在窗体中混合使用的测试说明如表 17-20 所示。

表 17-20　对各种控件在窗体中混合使用的测试说明

测试内容	测试方法与步骤	测试判断准则
控件间的相互作用	从上到下、从左到右观看、操作	正确
Tab 键的顺序	观看、操作	正确
热键的使用	观看、操作	正确
Enter 键和 Esc 键的使用	观看、操作	正确

20. 查找替换

对查找替换的测试说明如表 17-21 所示。

表 17-21　对查找替换的测试说明

测试项	测试内容	测试方法与步骤	测试判断准则
查找	直接查找	观看、操作	正确
	查找部分	观看、操作	正确
	查找全部	观看、操作	正确
	在组合框中寻找已经查找过的内容	观看、操作	正确
替换	直接替换	观看、操作	正确
	替换部分	观看、操作	正确
	替换全部	观看、操作	正确
	关闭“查找/替换”对话框，不执行任何操作，直接退出	观看、操作	正确

21. 插入操作

对插入操作的测试说明如表 17-22 所示。

表 17-22　对插入操作的测试说明

测试内容	测试方法与步骤	测试判断准则
插入文件	观看、操作	正确
插入图像	观看、操作	正确
在文档中插入文档本身	观看、操作	正确

（续表）

测试内容	测试方法与步骤	测试判断准则
移除插入的源文件	观看、操作	正确
更换插入的源文件的内容	观看、操作	正确
插入链接文件	观看、操作	正确
在文档中链接文档本身	观看、操作	正确
移除插入的源文件	观看、操作	正确
更换插入的源文件的内容	观看、操作	正确
插入程序允许的对象（如在 Word 中插入 Excel 工作表）	观看、操作	正确
修改所插入对象的内容	观看、操作	正确
卸载生成插入对象的程序	观看、操作	正确

22. 编辑操作

对编辑操作的测试说明如表 17-23 所示。

表 17-23　对编辑的测试说明

测试项	测试内容	测试方法与步骤	测试判断准则
剪切操作	对文本进行剪切	观看、操作	正确
	对文本框进行剪切	观看、操作	正确
	对图文框进行剪切	观看、操作	正确
	剪切图像	观看、操作	正确
	文本图像混合剪切	观看、操作	正确
复制操作	对文本进行复制	观看、操作	正确
	对文本框进行复制	观看、操作	正确
	对图文框进行复制	观看、操作	正确
	复制图像	观看、操作	正确
	文本图像混合复制	观看、操作	正确
粘贴操作	粘贴剪切的文本	观看、操作	正确
	粘贴剪切的文本框	观看、操作	正确
	粘贴剪切的图文框	观看、操作	正确
	粘贴所剪切的图像	观看、操作	正确
	剪切后在不同的程序中粘贴	观看、操作	正确
	多次粘贴同一内容	观看、操作	正确
	利用粘贴操作强制输入程序所不允许输入的数据	观看、操作	正确

23. 窗体

对窗体的测试说明如表 17-24 所示。

表 17-24　对窗体的测试说明

测试内容	测试方法与步骤	测试判断准则
窗体大小要合适，控件布局要合理	观看、操作	合理
快速或慢速移动窗体，背景及窗体本身刷新必须正确	观看、操作	正确
缩放窗体，窗体上的控件应随窗体的大小变化而变化	观看、操作	变化
显示分辨率，必须在不同的分辨率的情况下测试程序的显示是否正常	观看、操作	正常
状态栏是否显示正确	观看、操作	正确
工具栏的图标执行操作是否有效	观看、操作	有效
是否与菜单栏中的图标显示一致	观看、操作	一致
错误信息内容是否正确	观看、操作	正确

24. 控件

对控件的测试说明如表 17-25 所示。

表 17-25　对控件的测试说明

测试内容	测试方法与步骤	测试判断准则
控件的字体和大小要一致	观看、操作	一致
注意全角、半角混合	观看、操作	正确
中英文混合	观看、操作	不存在

25. 菜单

对菜单的测试说明如表 17-26 所示。

表 17-26　对菜单的测试说明

测试内容	测试方法与步骤	测试判断准则
菜单是否可以正常工作	观看、操作	正常工作
菜单是否与实际执行内容一致	观看、操作	一致
是否有错别字	观看	不存在错别字
快捷键是否重复	观看、操作	不存在重复
热键是否重复	观看、操作	不存在重复
快捷键与热键操作是否有效	观看、操作	有效
是否存在中英文混合	观看、操作	不存在
是否存在不同权限的用户登录一个应用程序	观看、操作	不存在
是否存在不同级别的用户可以看到不同级别的菜单并使用不同级别的功能	观看、操作	不存在
鼠标右键快捷菜单是否有效	观看、操作	有效

17.3　登录、添加、删除、查询模块测试用例设计实例

登录、添加、删除、查询是软件与用户交互的最直接的模块，测试用例设计如下。

- 测试目的：检验登录、添加、删除、查询模块的可用性。检验登录、添加、删除、查询模块可靠性。
- 测试对象：本次测试的对象为登录、添加、删除、查询模块。
- 测试内容：登录、添加、删除、查询模块。

1. 登录模块

对登录模块的测试说明如表 17-27 所示。

表 17-27　对登录模块的测试说明

测试内容	测试方法与步骤	测试判断准则
用户名符合要求	观看、操作	符合正确要求
密码符合要求	观看、操作	符合正确要求
用户名和密码都符合要求	观看、操作	符合正确要求
用户名不符合要求	观看、操作	不符合要求
密码不符合要求	观看、操作	不符合要求
用户名和密码都不符合要求	观看、操作	不符合要求
用户名为空	观看、操作	不符合要求
密码为空	观看、操作	不符合要求
用户名、密码都为空	观看、操作	不符合要求
数据库中存在的用户名	观看、操作	符合要求
数据库中存在的密码	观看、操作	符合要求
数据库中不存在的用户名	观看、操作	不符合要求
数据库中不存在的密码	观看、操作	不符合要求
数据库中存在的用户名，错误的密码	观看、操作	符合要求
数据库中不存在的用户名，存在的密码	观看、操作	不符合要求
输入的数据前存在空格	观看、操作	不符合要求
输入正确的用户名、密码以后按 Enter 键是否能登录	观看、操作	符合要求

2. 添加模块

对添加模块的测试说明如表 17-28 所示。

表 17-28　对添加模块的测试说明

测试内容	测试方法与步骤	测试判断准则
添加的数据项均合理，检查数据库中是否添加了相应的数据	观看、操作	添加相应数据
留出一个必填数据为空	观看、操作	不符合要求
添加的数据项不符合要求的地方要有错误提示	观看、操作	有错误提示
是否支持 Table 键	观看、操作	支持
按 Enter 键后是否能保存	观看、操作	能保存

3. 删除

对删除模块的测试说明如表 17-29 所示。

表 17-29　对删除模块的测试说明

测试内容	测试方法与步骤	测试判断准则
删除一个数据库中存在的数据	观看、操作	查看数据库中是否删除
删除一个数据库中并不存在的数据	观看、操作	是否有错误提示
输入一个格式错误的数据	观看、操作	是否有错误提示
输入的正确数据前加空格	观看、操作	是否能正确删除数据前的空格
什么也不输入	观看、操作	是否有删除

4. 查询

对查询模块的测试说明如表 17-30 所示。

表 17-30　对查询模块的测试说明

测试内容	测试方法与步骤	测试判断准则
输入的查询条件为数据库中存在的数据	观看、操作	是否能正确查出相应的数据
输入正确的查询条件以前加上空格	观看、操作	是否能正确查出相应的数据
输入格式或范围不符合要求的数据	观看、操作	是否有错误提示
输入数据库中不存在的数据	观看、操作	是否有错误提示
不输入任何数据	观看、操作	是否有错误提示
模糊查询	观看、操作	在查询的基础上加上一些字符，看是否能查出数据库中所有的相关信息

17.4　宽带接入网网络管理系统测试用例设计实例

测试目的：检验网络管理系统功能的可用性、检验网络管理系统性能的可靠性、检验用户界面是否友好。

测试对象：本次测试的对象为宽带接入网网络管理系统（不含代理软件部分）。

测试内容：系统配置、拓扑管理、网络故障功能、网络性能管理功能。

1. 系统配置

系统配置包括系统管理域配置、系统刷新控制、系统创建控制、创建网络地图、启动系统轮询等操作功能。对系统配置的测试说明如表 17-31 所示。

表 17-31　对系统配置的测试说明

测试项	测试内容	测试方法与步骤	测试判断准则
系统管理域配置	测试网管软件配置管理范围的功能	选择“系统配置”|“管理域配置”命令；设定轮询范围，地址类型：IPv4；地址：162.168.88.0；掩码：255.255.255.0；网络类型：以太网；轮询间隔：180；单击“确定”按钮	输入网络 IP 地址和掩码，检查配置是否成功有效。若出错，是否有出错原因报告，报告是否准确

（续表）

测试项	测试内容	测试方法与步骤	测试判断准则
系统刷新控制	测试网管软件系统刷新控制的功能	选择“系统配置”|“系统刷新控制”命令，设定“空闲状态”，单击“查询”按钮，确认配置正确	检查是否轮询性能数据。若出错，是否有出错原因报告，报告是否准确
系统创建控制	测试网管软件系统创建控制功能	选择“系统配置”|“系统创建控制”命令，单击“创建”按钮，等待创建进程；创建完成后，单击“确认”按钮	查看子网连接表、子网表、设备表是否将该网络的拓扑信息保存
创建网络地图	测试网管软件系统创建网络地图功能	选择“系统配置”|“创建网络地图”命令；单击“确认”按钮	查看子网连接表、子网表、设备表是否将该网络的拓扑信息保存
启动系统轮询	测试网管软件系统轮询控制的功能	选择“系统配置”|“系统刷新控制”命令；设定“轮询状态”，单击“查询”按钮，确认是否正确	检查是否轮询性能数据。若出错，是否有出错原因报告、报告是否准确

2．拓扑管理

测试网管软件的拓扑配置、搜索、刷新和拓扑操作的功能。对拓扑管理的测试的说明如表 17-32 所示。

表 17-32　对拓扑管理的测试说明

测试项	测试内容	测试方法与步骤	测试判断准则
子网刷新功能	拓扑刷新功能	选定子网；选择“拓扑管理”|“刷新本子网”命令	查看子网拓扑是否刷新
刷新全部网络功能	拓扑刷新功能	选择“拓扑管理”|“创建网络地图”命令；单击“确定”按钮	查看整个网络拓扑是否刷新
添加设备	在拓扑图中添加设备的功能	选择“拓扑管理”|“添加设备”命令；单击“确定”按钮，添加虚拟设备	在拓扑图中添加虚拟设备
设备属性编辑	编辑设备属性	选定添加的虚拟设备；单击右键，在弹出的快捷菜单中选择“设备属性”命令；配置 Basic Attributes 、配置 IP Config Attributes、配置 IP Interface、配置 MAC Interface、配置 Port Interface	在拓扑图中设备属性配置是否成功
删除设备	在拓扑图中删除设备的功能	选择“拓扑管理”|“添加设备”命令；单击“确定”按钮，添加虚拟设备。选定该虚拟设备；选择“拓扑管理”|“删除设备”命令	在拓扑图中删除设备是否成功

（续表）

测试项	测试内容	测试方法与步骤	测试判断准则
设备升级	在拓扑图中设备升级到它的上一级子网的功能	选定设备；选择“拓扑管理”|“设备升级”命令；单击“确定”按钮	设备升级到它的上级子网是否成功
设备降级	在拓扑图中设备降级到它的下一级子网的功能	选定设备；选择“拓扑管理”|“设备降级”命令；选定降级到的子网；单击“确定”按钮	设备降级到它的下级子网是否成功
添加连接	在拓扑图中添加两个设备间的连接功能	按 Ctrl 键和选定的两个设备;选择“拓扑管理”|“添加连接”命令；单击“确定”按钮	添加设备连接是否成功
删除连接	在拓扑图中删除两个设备间的连接功能	选定连接；选择“拓扑管理”|“删除连接”命令；单击“确定”按钮	删除设备连接是否成功
添加子网	在拓扑图中添加子网的功能	选择“拓扑管理”|“添加子网”命令；单击“确定”按钮	子网添加是否成功
删除子网	在拓扑图中删除子网的功能	选择“拓扑管理”|“删除子网”命令；单击“确定”按钮	子网删除是否成功
划分子网	测试在一个子网中，以掩码长度为准，子网进一步划分的功能	选定子网；选择“拓扑管理”|“划分子网”命令；网络 IP 为 162.168.88.240、掩码为 255.255.255.240，单击“确定”按钮	子网划分是否成功有效
汇聚子网	测试将几个子网融合成一个子网，包括这几个子网中的所有网元的功能	选定子网；选择“拓扑管理”|“汇聚子网”命令；单击“确定”按钮	子网汇聚是否成功
设备属性查询	测试设备属性查询功能	选定设备；右击鼠标，在弹出的快捷菜单中选择“属性信息”命令，查询选定设备的基本配置信息；右击鼠标，在弹出的快捷菜单中选择“当前故障信息”命令，查询选定设备的故障信息；右击鼠标，在弹出的快捷菜单中选择“故障日志”命令，查询选定设备的故障日志	设备属性查询是否成功
视频服务器的属性查询	测试视频服务器频道管理和频道故障报告功能	选定视频服务器；右击鼠标，在弹出的快捷菜单中选择“频道列表”命令，查询当前频道的状态信息；右击鼠标，在弹出的快捷菜单中选择“频道状态示意图”命令，查询频道故障信息	视频服务器的属性查询是否成功

3．网络故障功能

测试网管软件的故障检测、分析归并和报警功能。

测试配置：建立一个 LAN，其中主要包括安装了本网管软件和 SQL Server 数据库的 PC 机（服务器），安装了代理软件的被管网元 PC 机、家庭网关、交换机和路由器若干台。

测试准备：测试 LAN 正常，启动 PC 机（服务器）的网管软件，启动 PC 机、家庭网关、交换机和路由器中的代理软件。

对网络故障功能的测试说明如表 17-33 所示。

表 17-33　对网络故障功能的测试说明

测试项	测试内容	测试方法	测试判断准则
报警开关设置	测试网管软件启动/关闭故障管理的功能	选择“故障管理”|“报警开关”命令	查看故障管理功能启动是否正常
报警声音开关设置	测试网管软件启动/关闭故障管理声音报警功能	选择“故障管理”|“声音开关”命令	查看故障管理声音报警启动是否正常
故障报警配置	测试网管软件配置故障管理策略的功能	选择“故障管理”|“故障报警策略设置”命令，选定故障报警策略列表中故障级别为 5 级，报警声音关闭，进行报警策略设置	查看故障管理策略配置是否成功
当前故障设备列表	测试网管软件报告当前故障设备列表的功能	选择“故障管理”|“当前故障列表”命令	按当前故障设备排序，查看当前故障设备列表是否成功
当前故障详细列表	测试网管软件报告当前故障详细列表的功能	选择“故障管理”|“当前故障详细列表”命令	按故障发生时间排序，查看当前故障列表是否成功
故障日志查看	测试网管软件报告故障日志的功能	将一台 PC 机进行热启动，检查管理站是否收到 Trap 故障信息。关闭一台路由器端口，检查管理站是否受到 Trap 故障信息。在多台 PC 中广播大数据报>10M，引起广播风暴，检查管理站是否检测到流量性能超阀值信息。选择“故障管理”|“故障日志”命令	查看故障日志是否记录了被测故障事件
Trap 故障报告	测试网管软件接收被管网元发出的 Trap 信息的功能	将一台 PC 机进行冷启动，检查管理站是否收到 Trap 故障信息。关闭一台交换机端口，检查管理站是否收到 Trap 故障信息	查看是否报告 PC 机热启动和路由器端口断开的故障信息
轮询故障接收	测试网管软件接收轮询故障的功能	在一台 PC 中广播大数据报>10M，引起广播风暴，检查管理站是否检测到流量性能超阀值信息	查看性能故障链表是否报告接口丢包的性能故障
测试网管程序发现设备不通功能	以初始创建时设备表为依据，轮询各设备，发现不通就报警	管理域：162.168.88.0、255.255.255.0；被测试设备：162.168.88.81；分别运行创建程序和轮询程序；在系统正常轮询时断开被测试设备网线	发现设备 162.168.88.81 不通时进行故障报警

（续表）

测试项	测试内容	测试方法	测试判断准则
测试网管程序发现交换机端口断开功能	轮询交换机设备，发现端口不通就显示黄色报警	运行网络管理程序。管理域：162.168.88.0、255.255.255.0；被测试设备：162.168.88.57	断开被测试设备：162.168.88.57 的 9 号端口
故障归并	测试网管软件故障数据归并功能	查看数据库代理（SQL Server 代理）的作业中的故障归并。查看数据库，比较归并前后的故障数据	查看该作业是否成功。故障归并是否正确
设备故障属性显示	测试故障属性显示功能	选定一设备；单击鼠标右键，在弹出的快捷菜单中选择“当前故障详细信息”命令，单击“确定”按钮	查看是否显示当前故障设备的详细信息
设备故障属性显示	测试故障属性显示功能	选定一设备；单击鼠标右键，在弹出的快捷菜单中选择“故障日志”命令，单击“确定”按钮	查看是否显示当前故障设备的故障日志

4．网络性能管理功能

测试网管软件的性能轮询、性能数据压缩、实时性能报告、历史性能报告功能。

测试配置：建立一个 LAN，其中主要包括安装了本网管软件和 SQL Server 数据库的 PC 机（服务器），安装了代理软件的被管网元 PC 机、家庭网关、交换机和路由器若干台。

测试准备：测试 LAN 正常，启动 PC 机（服务器）的网管软件，启动 PC 机、家庭网关、交换机和路由器中的代理软件。

对网络性能管理功能的测试说明如表 17-34 所示。

表 17-34 对网络性能管理功能的测试说明

测试项	测试内容	测试方法	测试判断准则
实时性能报告	报告实时性能的功能	选定一设备；选择“性能管理”|“实时性能”命令；选择相关的性能和设备下标（如端口）；单击“查看性能图”按钮	检查实时性能报告显示
历史性能报告	报告历史性能的功能	选定一设备；选择“性能管理”|“历史性能”命令；选择相关的性能和设备下标（如端口）；选定时间范围；单击“查看历史图”按钮	检查历史性能报告显示
性能定义	测试性能定义功能	选择“性能管理”|“性能定义”命令，选择性能对象进行编辑。单击“编辑公式”按钮进入公式编辑器，具体编辑该性能对象的产生式	性能定义成功
性能故障定义	测试性能故障定义功能	100～1000KB 为二级故障；10～100KB 为三级故障；1～10KB 为四级故障；<1KB 为五级故障	性能故障定义成功

（续表）

测试项	测试内容	测试方法	测试判断准则
性能数据压缩	测试网管软件将实时性数据按一定的粒度进行压缩的功能	查看数据库代理（SQL Server代理）的作业中的性能数据压缩。查看数据库，比较压缩前后的性能数据	查看该作业是否成功，数据压缩是否正确

17.5　某部电子政务应用平台测试用例设计实例

某电子政务系统是针对某实际业务流程设计的公文处理、档案管理及公文档案的转换等主要业务流程，同时包括资料管理、会务管理、联络工作、行政管理等模块，囊括了所有业务工作，实现了真正意义上的电子政务。

该系统是某电子政务主体系统，该系统和某电子政务辅助决策系统、消费卡管理系统、门禁管理系统共同组成某电子政务系统，结构如图 17-1 所示。

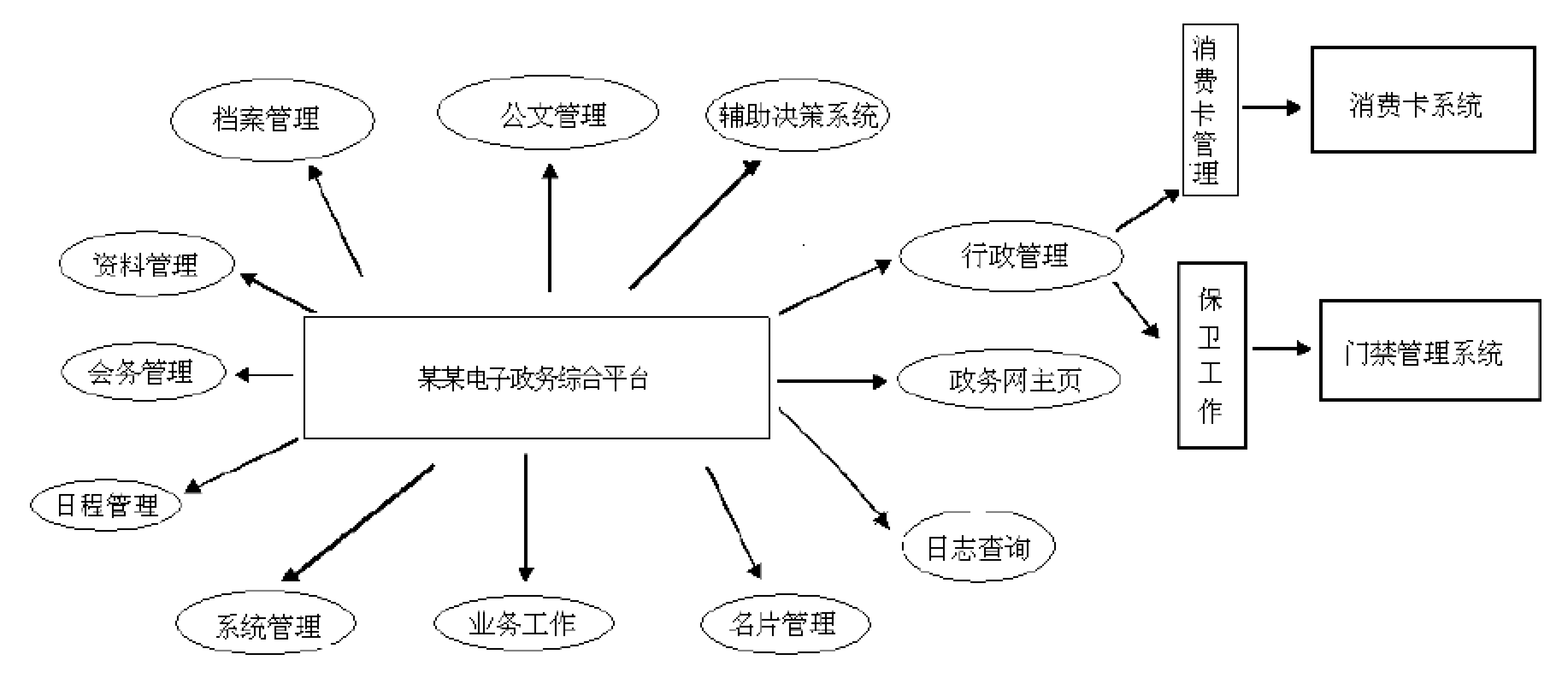


图 17-1　某电子政务工程综合电子政务应用平台

17.5.1　主页信息发布测试用例设计实例

测试目的：检验电子政务应用平台。

测试对象：本次测试的对象为电子政务应用平台。

测试内容：主页信息发布。

主页信息发布测试用例设计实例如表 17-35 所示。

表 17-35　主页信息发布测试用例设计实例

测试项	测试内容	测试方法与步骤	测试判断准则
政务主页信息	页面查看	是否正确打开各个主页	正确
		单击树节点是否能正确打开各个主页	正确
		各个页面的显示内容是否和添加顺序一致	正确

（续表）

测试项	测试内容	测试方法与步骤	测试判断准则
信息发布维护	新建各个节点主页	新建节点信息，填写正确的网页地址，到信息发布界面查看，能够看到新建节点的主页	正确
	修改各个节点主页	修改节点信息，修改网页地址，到信息发布界面查看，能够看到页面更新	正确
	查询各个节点主页	选择查询条件，单击“查询”按钮，能够查询出符合条件的记录	正确
增加主页信息	增加信息发布页面超链接	能否增加信息发布页面超链接	正确
主页单位维护	查询	按照主页标识进行查询;按照主页名称进行查询	正确
	系统自动生成主页标识	观看、操作	正确
	主页修改功能	观看、操作	正确
	主页保存功能	观看、操作	正确
	主页删除记录功能	观看、操作	正确
授权管理	能否增加权限	观看、操作	正确
	能否删除权限	观看、操作	正确
复制/粘贴目录	是否能复制、粘贴目录	观看、操作	正确
	是否能添加目录	观看、操作	能正确添加目录
	是否能修改目录	观看、操作	能正确修改目录
	是否能删除目录	观看、操作	能正确删除目录

17.5.2 工作站设置测试用例设计实例

测试目的：检验工作站设置。

测试对象：本次测试对象为电子政务应用平台的工作站设置。

测试内容：工作站设置。

工作站设置测试用例设计实例如表 17-36 所示。

表 17-36 工作站设置测试用例设计实例

测试项	测试内容	测试方法与步骤	测试判断准则
工作站设置	对授权记录进行位置上下调整	观看、操作	正确
	增加授权记录	观看、操作	正确
	对授权记录进行修改	观看、操作	正确
	对已有授权记录进行删除	观看、操作	正确
	对已有授权记录进行保存	观看、操作	正确
	正常退出“授权设置”页面	观看、操作	正确
	清空内容	观看、操作	正确
	校验输入字段	观看、操作	正确
	选择各个对应字段的值	观看、操作	正确

（续表）

测试项	测试内容	测试方法与步骤	测试判断准则
工作站查询页面管理	输入站编号查询	观看、操作	正确
	输入房间号查询	观看、操作	正确
	输入 IP 查询	观看、操作	正确
	输入网卡号查询	观看、操作	正确
	新建工作站	观看、操作	正确
编辑工作站页面	新建记录	观看、操作	正确
	翻页显示记录	观看、操作	正确
	保存记录	观看、操作	正确
	删除记录	观看、操作	正确
	选择站编号	观看、操作	正确
	生成站编号	观看、操作	正确
	增加使用人员	观看、操作	正确
	删除使用人员	观看、操作	正确
	设置工作站设备权限状态	观看、操作	正确
	对输入字段进行校验	观看、操作	正确

17.5.3 文件维护测试用例设计实例

测试目的：检验文件维护。

测试对象：本次测试对象为电子政务应用平台的文件维护。

测试内容：文件维护。

文件维护测试用例设计实例如表 17-37 所示。

表 17-37 文件维护测试用例设计实例

测试项	测试内容	测试方法与步骤	测试判断准则
文件维护	能否对新建正常记录进行保存操作	观看、操作	正确
	文件的查询授权是否正确	观看、操作	正确
	文件查询	观看、操作	正确
	文件修改	观看、操作	正确
	本人或授权表中的人员是否能对文件进行修改、查询操作	观看、操作	正确
	各个按钮能否正常查询	观看、操作	正确
	排序方式选择是否正确	观看、操作	正确
	查询速度能否满足要求	观看、操作	正确
文件查询授权	文件的查询授权是否正确	观看、操作	正确
	本人或授权表中的人员是否能对文件进行修改操作	观看、操作	正确
	文件查询的各个按钮能否正常查询	观看、操作	正确
	连续单击“查询”按钮	观看、操作	正确

（续表）

测试项	测试内容	测试方法与步骤	测试判断准则
	文件查询的字段是否有输入校验	观看、操作	正确
	排序方式选择是否正确	观看、操作	正确
	查询速度能否满足要求	观看、操作	正确
	显示内容是否整齐	观看、操作	正确

17.5.4 查询显示页面测试用例设计实例

测试目的：检验查询显示页面

测试对象：本次测试对象为电子政务应用平台的查询显示页面。

测试内容：查询显示页面。

查询显示页面测试用例设计实例如表 17-38 所示。

表 17-38 查询显示页面测试用例设计实例

测试内容	测试方法与步骤	测试判断准则
显示内容是否整齐	观看、操作	正确
能否打开链接页面	观看、操作	正确
链接页面能否正确返回显示页面	观看、操作	正确
单击树节点是否能正确打开各个主页	观看、操作	正确
各个页面显示内容是否和添加顺序一致	观看、操作	正确

17.5.5 数据传输测试用例设计实例

测试目的：检验查询数据传输。

测试对象：本次测试对象为电子政务应用平台的数据传输。

测试内容：数据传输。

数据传输测试用例设计实例如表 17-39 所示。

表 17-39 数据传输测试用例设计实例

测试项	测试内容		测试方法与步骤	测试判断准则
能否正确登录系统	单击“登录”按钮能否弹出登录界面		观看、操作	正确
	填写密码不正确是否能给出提示		观看、操作	正确
数据传输系统主界面功能	工具菜单功能	状态栏功能是否正确	观看、操作	正确
		传输设置是否能弹出该界面进行设置	观看、操作	正确
		运行设置是否能够弹出该界面进行设置	观看、操作	正确
		管理查询是否能够弹出该界面进行设置	观看、操作	正确
		帮助设置是否能够弹出帮助窗口进行设置	观看、操作	正确
	主界面功能	隐藏服务按钮功能是否正常	观看、操作	正确
		手动传输按钮功能是否正常	观看、操作	正确
		清除记录按钮功能是否正常	观看、操作	正确
		退出系统按钮是否正常	观看、操作	正确

17.5.6　个人信息通信工具测试用例设计实例

测试目的：检验个人信息通信工具。

测试对象：本次测试对象为电子政务应用平台的个人信息通信工具。

测试内容：个人信息通信工具。

个人信息通信工具测试用例设计实例如表 17-40 所示。

表 17-40　个人信息通信工具测试用例设计实例

测试内容	测试方法与步骤	测试判断准则
能否正确添加目录	操作	正确
能否正确修改目录	操作	正确
能否正确删除目录	操作	正确
能否正确复制目录	操作	正确
是否能增加权限	操作	正确
是否能修改权限	操作	正确
能否能删除权限	操作	正确

17.5.7　公文管理测试用例设计实例

测试目的：检验公文管理。

测试对象：本次测试对象为电子政务应用平台的公文管理。

测试内容：公文管理。

公文管理测试用例设计实例如下。

1. 文件

对文件的测试说明如表 17-41 所示。

表 17-41　对文件的测试说明

测试内容	测试方法与步骤	测试判断准则
能否正确新建文件，要求：文件种类只可选择，不可输入；文件种类号在同一年度、同种文件中不能出现重复；局资料号在同一年度、同一个撰文单位中不能出现重复	观看、操作	正确
能否正确编辑文件	观看、操作	正确
能否正确修改文件	观看、操作	正确
能否正确复制文件	观看、操作	正确
能否正确修改文件的标题文字格式	观看、操作	正确
能否正确进行字处理	观看、操作	正确

2. 内部制文

对内部制文的测试说明如表 17-42 所示。

表 17-42　对内部制文的测试说明

测试内容	测试方法与步骤	测试判断准则
保持查询条件的缺省值，单击“查询”按钮，查询结果数与下级节点的查询结果是否一致	观看、操作	正确
重新登录系统，单击“内部制文”及以下各节点，查询结果应与个人设置中的默认查询期限一致	观看、操作	正确
在某节点下查询一次后，再次单击该节点时，仍显示上次查询的结果；编辑某条数据后，应按照编辑数据前的查询条件显示查询结果	观看、操作	正确
对所有查询条件组合查询，查询结果是否符合查询条件	观看、操作	正确
可以进行打印设置，并预览打印结果。没有报表模板的情况下给出中文提示信息；打印报表的字体设置应在改变字体设置的同时立即生效	观看、操作	正确
当选择了子节点时，单击“新建”按钮将打开与当前节点文件种类相同的制文页面，否则打开一个选择文件种类的页面	观看、操作	正确
如果已经输入文件数据，提示是否保存，否则直接关闭新建的文件页面，返回到查询结果页面，查询条件应与新建页面操作前相同	观看、操作	正确
单击“清空”按钮，是否能将所有查询条件设置为默认值	观看、操作	正确
当按钮文字为“高级查询”时，单击该按钮将显示更多查询条件；文字为“隐藏”时，单击该按钮将隐藏部分查询条件	观看、操作	正确

3. 公文草稿

对公文草稿的测试说明如表 17-43 所示。

表 17-43　对公文草稿的测试说明

测试内容	测试方法与步骤	测试判断准则
新建：页面初始化能否满足如下要求。 ● 密级：内部 ● 保存年限：无 ● 公文种类：公文草稿 ● 局登日前：系统当前日期 date ● 默认权限：本人修改 ● 撰文单位：操作员所属单位，局办文号和处办文号取“最大文号+1” ● 撰稿人：操作员姓名 ● 改变初始值：修改撰文单位，局办文号和处办文号是否相应改变 ● 不输入数据保存，应提示必须录入标题 ● 输入标题：中文与字母、数字组合，长度为 200，中文 100，字母及数字共 100	观看、操作	正确
编辑：选择已存在的记录，打开文稿编辑页面，显示文稿信息，进行如下授权。 ● 授权（部门拥有权/按级别授权）：给某部所有人员授予“阅读权”，然后给某部处级以上人员授“编辑权” ● 授权（个人拥有权）：选择“某部”时，将“领导组”和“秘书组”加入个人授权列表，其他人员不加入	观看、操作	正确

（续表）

测试内容	测试方法与步骤	测试判断准则
保存：在授权界面单击“保存”按钮，返回到文稿编辑界面后单击“退出”按钮，保存授权结果，不保存文稿编辑界面修改的信息	观看、操作	正确
复制增加，包括如下要求。 ● 进入文稿页面，单击“复制增加”按钮，局办文号、处办文号取最大文号+1，文稿其他信息（包括查询授权）与原文件相同，保存后复制增加的数据存入数据库 ● 进入文稿页面，单击“复制增加”按钮，局办文号、处办文号取最大文号+1，修改文号为被复制文件的文号，保存时应提示文号重复 ● 修改/复制增加公文草稿或其他内部制文后，保存、退出文稿编辑页面，可以看到刷新后的查询结果	观看、操作	正确
测试标题文字格式，包括字体、字号、颜色、行距、字间距等是否正确	观看、操作	正确
字处理，包括如下要求。 ● 测试正文文字格式，包括字体、字号、颜色。功能是否实现，是否满足用户要求 ● 正文文字对齐方式、缩进、编号是否正确 ● 测试正文行间距、字间距是否正确 ● 测试正文的复制、粘贴是否正确 ● 测试正文页眉、页脚的设置是否正确 ● 测试页面设置及上下左右页边距是否正确 ● 测试公文打印预览及打印功能是否正确 ● 测试正文文字的输入、编辑、删除功能是否正确 ● 文字编辑器底部信息栏是否正确	观看、操作	正确

4. 内部行文（部内请示）

对内部行文的测试说明如表 17-44 所示。

表 17-44　对内部行文的测试说明

测试内容	测试方法与步骤	测试判断准则
新建	观看、操作	正确
编辑	观看、操作	正确

5. 外部行文

对外部行文的测试说明如表 17-45 所示。

表 17-45　对外部行文的测试说明

测试内容	测试方法与步骤	测试判断准则
打印功能是否正常	观看、操作	正确
查询功能是否正常	观看、操作	正确
“承办单位”是否可正常选择或者填写	观看、操作	正确
选择“发文单位”下拉列表框是否正确	观看、操作	正确

（续表）

测试内容	测试方法与步骤	测试判断准则
选择“密级”、“紧急程度”下拉列表框是否正确	观看、操作	正确
“密级”、“紧急程度”下拉列表框的数据传递是否正确	观看、操作	正确
表格格式是否满足要求	观看、操作	正确
复制增加外部行文时，不复制相关办件	观看、操作	正确
“原件分发”功能是否正确	观看、操作	正确
“归档查询”功能是否正确	观看、操作	正确
“催办号码”功能是否正确	观看、操作	正确
是否能存入数据库	观看、操作	正确

6. 收文管理

对收文管理的测试说明如表 17-46 所示。

表 17-46　对收文管理的测试说明

测试内容	测试方法与步骤	测试判断准则
打印功能是否正常	观看、操作	正确
查询功能是否正常	观看、操作	正确
根据公文的密级，系统自动完成存取登记的过程	观看、操作	正确
根据公文的正文、拟办、批阅、阅办、归档、查询的不同域自动进行操作权限的读、写控制	观看、操作	正确
根据公文处理的收文程序，灵活控制定义公文传递流向	观看、操作	正确
收文拟办、批阅、主办、阅办流转时间控制、自动提示	观看、操作	正确
收文归档后的数据库实现多种结构、不同方式的查询	观看、操作	正确
收文管理与发文管理、档案管理的数据接口方式	观看、操作	正确
存入数据库	观看、操作	正确
收文时间：从 x 年 x 月 x 日至 x 年 x 月 x 日	观看、操作	正确
办结时间：从 x 年 x 月 x 日至 x 年 x 月 x 日	观看、操作	正确
催办，包括提醒登记（当有公文需要登记时）、提醒拟办（当有公文需要拟办时）、提醒批示（当有拟办好的公文，且设置了批示期限时）、提醒承办（当有公文需要承办时）、提醒回复（当有承办好的公文，且设置了办理期限时）	观看、操作	正确

7. 发文管理

对发文管理的测试说明如表 17-47 所示。

表 17-47　对发文管理的测试说明

测试内容	测试方法与步骤	测试判断准则
打印功能是否正常	观看、操作	正确
查询功能是否正常	观看、操作	正确
根据公文的密级，系统自动完成存取登记的过程	观看、操作	正确

（续表）

测试内容	测试方法与步骤	测试判断准则
根据公文的种类，系统自动完成单位内部和对外公文的起草、存取过程	观看、操作	正确
根据公文的类别，自动添加分配文号	观看、操作	正确
根据发文管理的程序，灵活控制公文的传递流向	观看、操作	正确
根据发文的审批、签发、文号、打印、盖章、发文的不同域自动进行操作权限的读、写控制	观看、操作	正确
根据发文的级别模拟电子公章的盖印、存入发文数据库并且实现多种结构、不同方式的查询	观看、操作	正确
根据发文管理自动存档作为永久备份	观看、操作	正确
公文处理系统要遵循国家行政机关公文格式国家标准	观看、操作	正确
正确存入数据库	观看、操作	正确

17.5.8 修改文件和修改撰文单位测试用例设计实例

测试目的：检验修改文件和修改撰文单位。

测试对象：本次测试对象为电子政务应用平台的修改文件和修改撰文单位。

测试内容：修改文件和修改撰文单位。

修改文件和修改撰文单位测试用例设计实例如下。

1. 修改文件测试用例设计实例

对修改文件测试用例设计实例的测试说明如表 17-48 所示。

表 17-48 对修改文件测试用例设计实例的说明

测试内容	测试方法与步骤	测试判断准则
发文号产生前，有文件编辑权的人都可以修改文种	观看、操作	正确
发文号产生后，有“修改文件权”的人可以修改文件名称	观看、操作	正确
本人修改	观看、操作	正确
本人修改+本局局、处领导修改	观看、操作	正确
本人修改+本局全体人员修改	观看、操作	正确
本人修改+本处领导修改	观看、操作	正确
本人修改+本处全体人员修改	观看、操作	正确

2. 修改撰文单位测试用例设计实例

对修改撰文单位测试用例设计实例的测试说明如表 17-49 所示。

表 17-49 对修改撰文单位测试用例设计实例的说明

测试内容	测试方法与步骤	测试判断准则
发文号产生前，有“文件编辑权”的人可以修改撰文单位	观看、操作	正确
发文号产生后，有“修改撰文单位权”的人可以修改撰文单位	观看、操作	正确
撰文单位需要修改原撰文单位所属局内处、科名称，否则局办文号、处办文号会发生跳号现象	观看、操作	正确

17.5.9　党、团、工会事务管理测试用例设计实例

测试目的：检验党、团、工会事务管理。

测试对象：本次测试对象为电子政务应用平台的党、团、工会事务管理。

测试内容：党、团、工会事务管理。

党、团、工会事务管理测试用例设计实例如表 17-50 所示。

表 17-50　党、团、工会事务管理测试用例设计实例

测试项	测试内容	测试方法与步骤	测试判断准则
党务管理	党组织管理	观看、操作登记和查询	正确
	党员管理	观看、操作登记和查询	正确
	申请入党人员管理	观看、操作登记和查询	正确
	党费管理	观看、操作登记和查询	正确
团务管理	团组织管理	观看、操作登记和查询	正确
	团员管理	观看、操作登记和查询	正确
	申请入团人员管理	观看、操作登记和查询	正确
	团费管理	观看、操作登记和查询	正确
工会管理	工会组织管理	观看、操作登记和查询	正确
	工会会员管理	观看、操作登记和查询	正确
	申请入工会人员管理	观看、操作登记和查询	正确
	工会费管理	观看、操作登记和查询	正确

17.5.10　贺电事务管理测试用例设计实例

测试目的：检验贺电事务管理。

测试对象：本次测试对象为电子政务应用平台的贺电事务管理。

测试内容：贺电事务管理。

贺电事务管理测试用例设计实例如下。

1. 接收贺电

对接收贺电的测试说明如表 17-51 所示。

表 17-51　对接收贺电的测试说明

测试内容	测试方法与步骤	测试判断准则
打印功能是否正常	观看、操作	正确
查询功能是否正确	观看、操作	正确
新建页面各个按钮文本框功能是否正常	观看、操作	正确
保存功能是否正常	观看、操作	正确

2. 发送贺电

对发送贺电的测试说明如表 17-52 所示。

表 17-52　对发送贺电的测试说明

测试内容	测试方法与步骤	测试判断准则
打印功能是否正常	观看、操作	正确
查询功能是否正确	观看、操作	正确
新建发送贺电功能是否正常	观看、操作	正确
其他功能是否正确	观看、操作	正确

3. 唁电

对唁电的测试说明如表 17-53 所示。

表 17-53　对唁电的测试说明

测试内容	测试方法与步骤	测试判断准则
打印功能是否正常	观看、操作	正确
文本框功能是否正常	观看、操作	正确
查询功能是否正确	观看、操作	正确
保存功能是否正常	观看、操作	正确

4. 贺卡

对贺卡的测试说明如表 17-54 所示。

表 17-54　对贺卡的测试说明

测试内容	测试方法与步骤	测试判断准则
打印功能是否正常	观看、操作	正确
查询功能是否正常	观看、操作	正确
文本框功能是否正常	观看、操作	正确
保存功能是否正常	观看、操作	正确
修改功能是否正常	观看、操作	正确

17.5.11　固定资产管理测试用例设计实例

测试目的：检验固定资产管理。

测试对象：本次测试对象为电子政务应用平台的固定资产管理。

测试内容：固定资产管理。

对固定资产管理测试用例设计实例的说明如表 17-55 所示。

表 17-55　对固定资产管理测试用例设计实例的说明

测试内容	测试方法与步骤	测试判断准则
资产输入功能	观看、操作	正确
资产统计功能	观看、操作	正确
资产查询功能	观看、操作	正确
资产修改功能	观看、操作	正确
资产打印功能	观看、操作	正确

（续表）

测试内容	测试方法与步骤	测试判断准则
按照资产的名称输入	观看、操作	正确
按照资产的原值输入	观看、操作	正确
按照资产的型号输入	观看、操作	正确
按照资产的折旧率输入	观看、操作	正确
按照固定资产的类别查询、打印和修改	观看、操作	正确
按照固定资产的名称查询、打印和修改	观看、操作	正确
按照固定资产的原值查询、打印和修改	观看、操作	正确
资产登记功能	观看、操作	正确
资产折旧功能	观看、操作	正确
资产管理功能	观看、操作	正确
年终报表	观看、操作	正确
万元以上资产登记生成处理	观看、操作	正确
千元以上资产登记生成处理	观看、操作	正确
百元以上资产登记生成处理	观看、操作	正确
万元以上资产折旧处理计算、更新数据	观看、操作	正确
千元以上资产折旧处理计算、更新数据	观看、操作	正确
百元以上资产折旧处理计算、更新数据	观看、操作	正确
折旧到期的资产处理更新、输出处理	观看、操作	正确
被折旧资产中途变卖处理更新、输出处理	观看、操作	正确
固定资产收回处理变更、合并处理	观看、操作	正确
固定资产再分配处理变更、分类处理	观看、操作	正确
固定资产损耗处理、输出处理	观看、操作	正确
折旧后万元以上固定资产一览表生成、输出处理	观看、操作	正确
折旧后千元以上固定资产一览表生成、输出处理	观看、操作	正确
折旧后百元以上固定资产一览表生成、输出处理	观看、操作	正确
折旧后百元以下固定资产一览表生成、输出处理	观看、操作	正确
注销功能是否正确	观看、操作	正确
删除功能是否正确	观看、操作	正确

17.5.12 会务管理测试用例设计实例

测试目的：检验会务管理。

测试对象：本次测试对象为电子政务应用平台的会务管理。

测试内容：会务管理。

对会务管理测试用例设计实例的说明如表 17-56 所示。

表 17-56 对会务管理测试用例设计实例的说明

测试内容	测试方法与步骤	测试判断准则
是否对正常记录保存	观看、操作	正确
左、右键是否屏蔽	观看、操作	正确
下拉框能否任意选择	观看、操作	正确
能否翻页	观看、操作	正确
能否复制增加	观看、操作	正确
能否对会议资料进行打印	观看、操作	正确
能否正确进行会议授权	观看、操作	正确
各个字段是否有校验	观看、操作	正确
能否对正常查询条件进行查询	观看、操作	正确
各字段是否有校验	观看、操作	正确

17.5.13 领导日程管理测试用例设计实例

测试目的：检验领导日程管理。

测试对象：本次测试对象为电子政务应用平台的领导日程管理。

测试内容：领导日程管理。

对领导日程管理测试用例设计实例的说明如表 17-57 所示。

表 17-57 对领导日程管理测试用例设计实例的说明

测试内容	测试方法与步骤	测试判断准则
新建功能：工具条功能是否正确、新建用户后能否自动刷新	观看、操作	正确
修改功能：修改后是否自动刷新、上下页翻页是否可用	观看、操作	正确
删除功能：能否删除记录、删除后能否自动刷新	观看、操作	正确
查询功能：单击“清空”按钮，是否能清空查询条件	观看、操作	正确
授权功能是否正确	观看、操作	正确
日期是否校验	观看、操作	正确
“上周”和“下周”按钮功能是否正确	观看、操作	正确
“查询授权”是否起作用	观看、操作	正确

17.5.14 机构管理维护测试用例设计实例

测试目的：检验机构管理维护。

测试对象：本次测试对象为电子政务应用平台的机构管理维护。

测试内容：机构管理维护。

对机构管理维护测试用例设计实例的说明如表 17-58 所示。

表 17-58　对机构管理维护测试用例设计实例的说明

测试内容		测试方法与步骤	测试判断准则
查询	能否输入单位名称进行查询	观看、操作	正确
	能否输入行政级别进行查询	观看、操作	正确
新增单位	能否正常打开“新增单位”窗口	观看、操作	正确
	能否不进行任何操作直接单击“关闭”按钮或者直接关闭窗口	观看、操作	正确
	输入“单位名称”时是否进行校验	观看、操作	正确
	输入“单位全称”时是否进行校验	观看、操作	正确
	输入“行政序列编码”时是否进行范围校验	观看、操作	正确
	填写次序在输入较大数时是否可正常保存	观看、操作	正确
	输入“备注”时是否进行校验	观看、操作	正确
	正常输入数据是否可新增单位并自动刷新	观看、操作	正确
已有单位	能否不进行任何操作直接单击“关闭”按钮或者直接关闭窗口	观看、操作	正确
	输入“备注”时是否进行校验	观看、操作	正确
	输入“单位描述”时是否正常	观看、操作	
	输入“单位说明”时是否可正常保存	观看、操作	正确
	输入“单位电话”时是否可正常保存	观看、操作	正确
新增人员	在“新增人员”中的选择“行政级别”与“机构人员编辑”中的选择“行政级别”是否一致	观看、操作	正确
	人员名称是否可以正常选择	观看、操作	正确
机构人员	人员名称是否可以修改，如不能修改是否给出提示	观看、操作	
	显示次序校验是否正确	观看、操作	正确
	人员简称是否正确	观看、操作	正确
	身份证号码是否校验	观看、操作	正确
	输入政治面貌是否正确	观看、操作	正确
	教育经历是否正确	观看、操作	正确
	删除时是否报错	观看、操作	正确
外单位管理	外单位能否使用单位序号和单位名称进行检索	观看、操作	正确
	外单位加入功能是否生效	观看、操作	正确
	前一页、下一页功能是否正常	观看、操作	正确
	信箱号是否正常（单位地址、单位全称）	观看、操作	正确

17.5.15　代码维护和主题词分类测试用例设计实例

测试目的：检验代码维护和主题词分类。

测试对象：本次测试对象为电子政务应用平台的代码维护和主题词分类。

测试内容：代码维护和主题词分类。

对代码维护测试用例设计实例的说明如表 17-59 所示。

表 17-59 对代码维护测试用例设计实例的说明

测试内容	测试方法与步骤	测试判断准则
是否能查询出符合条件的记录	观看、操作	正确
是否能正确新建代码维护，并保存	观看、操作	正确
新建功能是否正确	观看、操作	正确
修改功能是否正确	观看、操作	正确
删除功能是否正确	观看、操作	正确
是否能保存修改的代码维护	观看、操作	正确
是否能删除代码维护记录	观看、操作	正确

对主题词分类测试用例设计实例的说明如表 17-60 所示。

表 17-60 对主题词分类测试用例设计实例的说明

测试内容	测试方法与步骤	测试判断准则
是否能够查询出满足条件的记录	观看、操作	正确
是否能新建主题词并保存	观看、操作	正确
是否能保存修改主题词记录	观看、操作	正确
是否能删除主题词记录	观看、操作	正确
新建功能是否正确	观看、操作	正确
修改功能是否正确	观看、操作	正确
删除功能是否正确	观看、操作	正确

17.5.16 公文流转测试用例设计实例

测试目的：检验公文流转。

测试对象：本次测试对象为电子政务应用平台的公文流转。

测试内容：公文流转。

对公文流转测试用例设计实例的说明如表 17-61 所示。

表 17-61 对公文流转测试用例设计实例的说明

测试内容	测试方法与步骤	测试判断准则
对不同权限、不同角色实际模拟业务流程进行测试	观看、操作	正确
公文内制文系统能够在各种内制文模版中完成内制文的录入	观看、操作	正确
对不同角色、不同权限的人可通过设置权限查询受权实现灵活控制	观看、操作	正确
可以在政务系统网上完成公文审批和核文工作	观看、操作	正确
可以完成文件的归档和保存到个人资料库	观看、操作	正确
可以在政务网上完成整个办件的业务流程	观看、操作	正确
公文的打印	观看、操作	正确

17.5.17 目录测试用例设计实例

测试目的：检验目录。

测试对象：本次测试对象为电子政务应用平台的目录。

测试内容：目录。

对目录测试用例设计实例的说明如表 17-62 所示。

表 17-62　对目录测试用例设计实例的说明

测试内容	测试方法与步骤	测试判断准则
能否正确添加目录	观看、操作	正确
能否正确修改目录	观看、操作	正确
能否删除目录	观看、操作	正确
是否能增加权限	观看、操作	正确
能否删除权限	观看、操作	正确
能否复制/粘贴目录	观看、操作	正确

17.5.18　维护人员测试用例设计实例

测试目的：检验人员维护。

测试对象：本次测试对象为电子政务应用平台的人员维护。

测试内容：人员维护。

对人员维护测试用例设计实例的说明如表 17-63 所示。

表 17-63　对人员维护测试用例设计实例的说明

测试内容	测试方法与步骤	测试判断准则
查询功能是否正常	观看、操作	正确
能否新增人员	观看、操作	正确
权限分类：能否按权限标识查询、能否按权限名称查询、能否按权限标识与权限名称查询	观看、操作	正确
能否新建栏目、能否修改记录	观看、操作	正确
授权设置功能是否正确	观看、操作	正确
工作站权限设置是否正确	观看、操作	正确
能否查询页面、能否新建工作站、能否编辑工作站页面、能否新建记录	观看、操作	正确

17.6　电子政务应用平台主页功能测试用例设计实例

对电子政务应用平台主页功能测试用例设计实例的说明如表 17-64 所示。

表 17-64 对电子政务应用平台主页功能测试用例设计实例的说明

测试内容		测试方法与步骤	测试判断准则
打开各个主页	单击树节点是否能正确打开各个主页	观看、操作	正确
	各个页面的显示内容是否和添加顺序一致	观看、操作	
新建各个节点主页	新建节点，并填写正确的网页地址，到信息发布界面查看，能够看到新建节点的主页	观看、操作	正确
修改各个节点主页	修改节点信息，修改网页地址，到信息发布界面查看，能够看到页面更新	观看、操作	正确
查询各个节点主页	选择查询条件，单击“查询”按钮，能够查询出符合条件的记录	观看、操作	正确
填写代码生成器	正确填写“机器名”、“数据库名”、“表名”、“用户”、“密码”等，单击“提取数据”按钮，能否正确提取数据库字段名	观看、操作	正确
	错误填写 “机器名”、“数据库名”、“表名”、“用户”、“密码”等，单击“提取数据”按钮，能否给出提示	观看、操作	
生成文件	不填写关联字段，直接单击“生成”按钮能否给出提示	观看、操作	正确
	不填写标题，单击“生成”按钮能否给出提示	观看、操作	
	文件目录填写不正确，能否给出提示	观看、操作	
	对提取表的各个字段属性修改能否体现在生成文件中	观看、操作	
生成文件是否正确	生成的.aspx 页面内容是否和代码生成器的填写一致	观看、操作	正确
	生成.cs 文件中的变量、方法中的变量是否和代码生成器的填写一致	观看、操作	
能否正确登录系统	单击“登录”按钮能否弹出登录界面	观看、操作	正确
	填写密码不正确时是否给出提示	观看、操作	
工具菜单功能	状态栏功能是否正确	观看、操作	正确
	传输设置是否能弹出该界面进行设置	观看、操作	
	运行设置是否能够弹出该界面进行设置	观看、操作	
	管理查询是否能弹出该界面进行设置	观看、操作	
	帮助设置是否能够弹出帮助窗口进行设置	观看、操作	
主界面功能	“隐藏服务”按钮功能是否正常	观看、操作	正确
	“手动传输”按钮功能是否正常	观看、操作	
	“清除记录”按钮功能是否正常	观看、操作	
	“退出系统”按钮是否正常	观看、操作	

第18章　测试文档的写作

测试文档是对要执行的软件测试及测试的结果进行描述、定义、规定和报告的任何书面或图示信息。由于软件测试是一个很复杂的过程，同时也涉及到软件开发中其他一些阶段的工作，因此，必须把对软件测试的要求、规划、测试过程等有关信息和测试的结果，以及对测试结果的分析、评价，以正式的文档形式给出。

测试文档对于测试阶段工作的指导与评价作用是非常明显的。需要特别指出的是：在已开发的软件投入运行的维护阶段，常常还要进行再测试或回归测试，这时还会用到测试文档。测试文档的编写是测试管理的一个重要组成部分。

本章重点讨论以下内容：

- 测试文档的写作概述。
- 测试需求说明书写作的内容。
- 测试任务说明书写作的内容。
- 测试计划说明书写作的内容。
- 测试大纲写作的内容。
- 测试用例写作的内容。
- 缺陷跟踪报告写作的内容。
- 测试分析报告写作的内容。
- 程序错误报告写作的内容。
- 集成测试报告写作的内容。
- 单元测试报告写作的内容。
- 系统测试总结报告写作的内容。
- 验收测试报告写作的内容。

18.1　测试文档的写作概述

1. 测试文档的写作目的

测试文档写作是为提高测试工作的整体水平，对软件产品测试过程中发现的问题进行分析，为开发人员以后的修改、升级提供一个预防问题的依据，通过测试文档总结测试阶段的工作，并积累测试经验。

2. 测试文档的写作要求

测试文档的写作需要重点注意如下 7 点内容。

- 针对性：测试文档写作应分为不同对象、不同类型、不同层次，如对于面向管理人员和用户的文档，不应像开发文档（面向软件开发人员）那样过多地使用软件的专业术语；开发文档使用的专业词汇未被广泛认知的，应添加注释进行说明；缩写词未被广泛认知的，应在其后添加注释。
- 正确性：没有错字，漏字；文档间引用关系正确；图、表准确、清晰；文档细节正确。
- 准确性：意思表达准确清晰，没有二义性；避免使用比较性词语，如提高，应定量说明提高程度；避免使用模糊、主观的术语，减少不确定性，如界面友好、操作方便；正确使用标点符号，避免产生歧义。
- 完整性：意思表达完整，语句要通顺；不遗漏软件测试的要求和有关信息。
- 简洁性：尽量不要采用较长的句子来描述，无法避免时，应注意使用正确的标点符号；力求简洁明了，每个意思只在文档中表达一次；句子简短完整，具有正确的语法、拼写和标点；每个陈述语句只表达一个意思。
- 统一性：统一采用专业术语和项目规定的术语集；同一个意思和名称，前后描述的用语要一致；文档前后使用的字体要统一。
- 易读性：文字描述要通俗易懂；前后文关联词使用恰当；文档变更内容使用不同颜色与上个版本区别开来。

3. 测试文档的写作内容

测试文档写作的内容如图 18-1 所示。

文档项目		所属机构				编号				
文档题目	编写人	日期	页数	版本	审核	日期	评审	日期	批准	日期
1. 测试任务说明书 2. 测试计划说明书 3. 测试大纲 4. 测试用例 5. 缺陷跟踪报告 6. 测试分析报告 7. 程序错误报告 8. 集成测试报告 9. 单元测试报告 10. 系统测试总结 11. 验收测试报告										
外部		内部	★	机密		高级机密				
备注										

图 18-1　测试文件内容

18.2 测试需求说明书写作的内容

18.2.1 测试需求说明书的写作方法

测试需求说明书阐述一个测试软件系统必须提供的功能、性能以及它所要考虑的限制条件，它不仅是系统测试和用户文档的基础，也是所有子系列项目规划、设计和编码的基础。它应该尽可能完整地描述系统预期的外部行为和用户可视化行为。除了设计和实现上的限制外，软件需求规格说明不应该包括设计、构造、测试或工程管理的细节。

测试需求说明书写作有如下三种：

- 利用结构化和自然语言编写文本型文档。
- 建立图形化模型，这些模型可以描绘转换过程、系统状态和它们之间的变化、数据关系、逻辑流或对象类和它们的关系。
- 编写形式化规格说明，参照开发文档模板编写测试需求说明书。模板为软件测试记录功能需求和各种其他与需求相关的重要信息提供了统一的结构。

18.2.2 测试需求说明书的写作模板

测试需求说明写作模板是很有用的，但有时要根据项目特点进行适当的改动。很多人拿需求说明书的模板套用，这就有很大的风险，例如会出现需求不全、需求范围界定不到位、需求分类不明确等因素。

测试需求说明写作模板的主要内容包括如下 15 点。

1. 引言

引言提出了对软件测试需求规格说明的纵览，这有助于读者理解文档如何编写并且如何阅读和解释。

（1）编写目的

对测试产品进行定义，阐述编写测试需求说明书的目的及意义，说明编写这份软件需求说明书的目的，指出预期的读者。在该文档中详尽说明了这个产品的软件需求，包括修正或发行版本号。如果这个软件测试需求规格说明只与整个系统测试的一部分有关系，那么就只定义文档中说明的部分或子系统测试。

（2）测试项目背景

对测试项目背景的说明如下：

- 需要阐述测试项目的软件系统的名称。
- 填写本项目的测试任务提出者、开发者、用户。
- 说明测试该软件系统同其他系统或其他机构的基本的相互来往关系。

（3）定义

列出测试需求说明书中用到的专门术语的定义和外文首字母词组的原词组、缩写词和符号。

（4）文档约定

开发人员、项目经理、营销人员、用户、测试人员或文档的编写人员描述了文档中剩余部分的内容及其组织结构，提出了最适合于每一类型读者阅读文档的建议，描述编写文档时所采用的标准或排版约定，包括正文风格、提示区或重要符号，列出进行本软件测试工作的约束，例如经费限制、测试期限、设备条件、用户的资料准备和交流上的问题等。

（5）产品的测试范围

需要简述产品的测试范围。

（6）参考文献

参考文献列出了测试需求说明书中引用和参考的资料，如本项目经核准的计划任务书、合同、上级机关的批文；属于本项目的其他已发表的文件；测试需求说明书中引用的文件、资料，包括所要用到的软件开发标准；列出这些文件资料的标题、文件编号、发表日期和出版单位，说明能够得到这些文件资料的来源。

2. 测试任务概述

（1）测试目标

测试目标描述了要测试产品应达到的目标，包括软件组成、硬件组成、网络构成、系统架构及其说明等。

（2）运行环境

运行环境描述了软件测试的运行环境，包括硬件平台、操作系统和版本，还有其他的软件组件或与其共存的应用程序，如测试硬件环境、测试软件环境。

（3）条件与限制

对测试条件与限制的说明如下。

- 测试实现上的限制：测试应遵守的条件和受到的限制，主要有如下几方面：开发单位或部门应具备的条件；开发者完成开发工作的期限；应用环境受到的限制，如网络带宽；可维护性、可移植的限制；软件使用者、管理者对计算机了解的限制。
- 确定影响测试人员自由选择的问题，并说明这些问题为什么成为一种限制。
- 确定测试项目对外部因素存在的依赖。

3. 系统特性

对系统特性的说明如下。

- 说明和优先级：对系统特性简短说明并指出优先级是高、中、低。
- 评价：对系统利益、损失、费用、风险和优先级进行评价。相对优先等级可以从 1（低）到 9（高）。
- 响应序列：列出用户动作、来自外部设备的信号输入、定义系统响应序列。序列将与使用实例相关的对话元素相对应。

4. 数据的一致性、正确性测试

对数据的一致性、正确性测试，需要测试静态数据、动态数据、数据库、数据流图、数据字典。

5. 用例描述

用例描述是以文本的方式描述每一个用例中角色与系统相互交互的规格说明，如表 18-1 所示。

表 18-1　用例描述

描述对象	描述内容
标识符	用例的惟一标识符
说明	对用例的概要说明
参与者	与该用例相关的参与者列表以及参与者的特点
频度	参与者访问此用例的频率

6. 功能测试要求

功能测试要求将列出与该特性相关的详细功能需求，这些是必须提交给用户的软件功能，使用户可以使用所提供的特性执行服务或者使用所指定的实例执行任务。功能测试要求描述产品如何响应可预知的出错条件、非法输入或动作，必须惟一地标识每个需求。对功能测试要求的说明如下。

- 功能划分。
- 功能描述。
- 功能测试用例描述。
- 测试非功能需求。
- 测试输入输出要求。
- 测试数据管理能力要求。
- 测试安全保密性要求。
- 测试灵活性要求。

7. 性能需求测试要求

性能需求测试要求包括测试精度、时间特性、适应性等要求。

8. 运行测试要求

对运行测试要求的说明如下。

- 压力测试。
- 用户界面，包括描述对字体大小的测试、按钮大小的测试、窗口大小的测试、按钮标题大小的测试、页面标题的测试、页面设置的测试、编辑框的测试、快捷键的校验的测试、导航条的测试。
- 硬件接口：描述系统中软件和硬件每一接口的特征。这种描述可能包括支持的硬件类型、软硬件之间交流的数据、控制信息的性质以及所使用的通信协议。
- 软件接口：描述该产品与其他外部组件（由名字和版本识别）的连接，包括数据库、操作系统、工具、库和集成的商业组件，并描述在软件组件之间交换数据或消息的目的、所需要的服务以及内部组件通信的性质，确定将在组件之间共享的数据。

- 通信接口：描述与产品所使用的通信功能相关的需求，包括电子邮件、Web 浏览器、网络通信标准或协议及电子表格等，定义了相关的消息格式，规定通信安全或加密问题、数据传输速率和同步通信机制，例如描述计算机与机器硬件接口、波特率等的测试；通信过程中断电的测试；人为中断通信的测试；连续多次通信的测试；通信过程中随意操作按钮键的测试。
- 设备：列出运行该软件所需要的设备，说明其设备及其专门功能。
- 故障处理：列出可能的软件、硬件故障以及对各项性能而言所产生的后果和对故障处理的要求。

9. 安全测试要求

对安全测试要求的说明如下。

- 安全设施测试需求：详尽陈述与产品使用过程中可能发生的损失、破坏或危害相关的需求。定义必须采取的安全保护或动作，还有那些预防的潜在的危险动作。明确产品必须遵从的安全标准、策略或规则。
- 安全性测试需求：详尽陈述与系统安全性、完整性或与私人问题相关的需求，这些问题将会影响到产品的使用和产品所创建或使用的数据的保护。定义用户身份确认或授权需求。明确产品必须满足的安全性或保密性策略。

10. 文件传输

对文件传输测试的说明如下：

- 描述文件上传到 FTP 或 Server 服务器端后，系统是否会死机的测试。
- 描述下载文件、上传文件成功与否的反馈信息的测试。
- 描述 FTP 中大量文件下载的测试。
- 描述同时下载、上传多个文件的测试。

11. 数据导入导出测试

对数据导入导出测试的说明如下：

- 多条记录导入的测试。
- 多个文件导入数据库的测试。
- 导入文件名的测试。
- 导入多条完全相同的基础数据的测试。
- 导入非法数据是否有校验的测试。

12. 安装测试

对安装测试的说明如下：

- 系统安装运行的测试。
- IE 浏览器版本的测试。
- 运行之前各个环境设置的测试。
- 部分运行程序文件被删除的测试。

13. 回归测试

测试客户提出来的 Bug 或需求之前，必须先看一下这些 Bug 与需求的修改会影响到哪些地方、哪些页面、哪些功能，然后针对这些影响项进行测试，从而使测试做到更加全面，以免漏掉有些地方没有测试到。

14. 用户文档测试

用户文档测试列举出将与软件一同发行的用户文档部分，例如用户手册、在线帮助和教程，明确所有已知的用户文档的交付格式或标准。

15. 其他专门要求

其他专门要求主要包括：用户单位对使用方便的要求，以及对可维护性、可补充性、易读性、可靠性、异常处理、运行环境可转换性的特殊要求等。

18.3 测试任务说明书写作的内容

测试任务说明书是经理或开发项目的负责人编写的，传递给软件测试人员、软件开发人员、软件管理人员。

18.3.1 测试任务、测试质量和测试范围

从用户的角度出发，测试实施任务和时间人员安排，其中软件测试人员、软件开发人员不能影响测试进度，并对软件的开发过程中每个版本完成测试任务。

1. 测试任务

需要确保所有的测试人员都知道项目和系统的目标，对测试人员的测试任务的说明如下：

- 测试人员了解自己需要做什么，有一个清晰的认识。
- 测试项目负责人了解自己需要做什么，有一个清晰的认识。
- 软件管理人员了解自己需要做什么，有一个清晰的认识。
- 测试小组了解系统是做什么，有一个清晰的认识。

2. 测试质量

测试质量应该包括产品的测试质量和测试小组的测试质量，这将关系到系统的功能或性能是否正常，必须做到以下几点：

- 测试质量确认。
- 所有的测试案例已经执行过。
- 所有的自动测试脚本已经执行通过。
- 所有的重要等级的 Bug 已经解决并由测试验证。
- 每一部分的测试已经被确认完成。
- 重要的功能没有 Bug。

3. 测试范围

对测试范围的说明如下。

- 流程测试：流程测试采用业务流程、数据流程、逻辑流程来检查软件是否能够按流程操作时正确处理。
- 边界值测试：边界值测试用于对边界数据进行测试，确保系统功能正常，程序无异常。
- 容错性测试：容错性测试用于检查系统的容错能力，错误的数据输入不会对功能和系统产生非正常的影响，程序对错误的输入有正确的提示信息。
- 异常测试：异常测试用于检查系统能否处理异常。
- 安装测试：安装测试用于检查系统是否能正确安装、配置。
- 易用性测试：易用性测试用于检查系统是否易用、友好。
- 界面测试：界面测试用于检查界面是否美观合理。
- 接口测试：接口测试用于检查系统是否能与外部接口正常工作。
- 配置测试：配置测试用于检查配置是否合理、正常。
- 性能测试：性能测试用于提取系统性能的数据，检查系统是否满足在需求中所规定达到的性能。
- 压力测试：压力测试用于检查系统是否能承受大压力，测试产品应该能够在高强度条件下正常运行，并不会出现任何错误。
- 兼容性测试：兼容性测试对于 C/S 架构的系统来说，需要考虑客户端支持的系统平台；对于 B/S 架构的系统来说，需要考虑用户端浏览器的版本。
- 升级测试：升级测试用于进行专门的割接测试或升级测试，提供工程升级割接方案。
- 功能测试：根据系统需求文档和设计文档，检查产品是否正确实现了功能。
- 单元测试。
- 集成测试。
- 系统测试。
- 回归测试：回归测试用于检查程序修改后有没有引起新的错误、是否能够正常工作以及能否满足系统的需求。
- 验收测试。
- 文档测试：文档测试用于检查文档是否足够、描述是否合理。

18.3.2 确定测试进度和管理

1. 确定测试进度

制定和确定测试进度时，必须由开发人员和相关的测试部门人员共同进行。在制定测试进度时，必须考虑到合理地配置测试资源（测试设备、测试所需要的技术文档资料、测试人员和对人员进行必须的培训）。为了使所制定的测试进度正常有效，必须对其所制定的测试进度加以量化。要制定测试的各个阶段的测试进度。有特殊情况时还必须制定特定系统的测试进度，如文件管理系统、资料库内容功能测试等。所制定的测试进度中必须含有修改问题和复查的时间。

2. 管理

可以根据测试进度划分的测试阶段时间进行管理监控、根据测试大纲和测试用例过程执行的情况来管理测试进度、通过测试通知单的平均反馈时间和更改程序的速度来管理测试进度、通过问题趋势图来管理各阶段的测试进度。

18.3.3 测试注意事项

根据《软件开发规范》应仔细检查以下内容：

- 软件的界面是否合乎要求（每一个子界面也应如此），其中，应注意提示信息和软件开发商信息是否正确。
- 小的图标是否合乎要求。检查菜单中的各项功能和功能按钮是否能正确使用。
- 根据《软件开发规范》、《用户需求》及《软件详细设计》设计测试用例（以边界值法、等价类划分法为主）。
- 对功能界面要求注意与功能相关的信息显示及显示位置是否正确。数据输入界面应注意文字格式、数字和文字的区别。
- 是否能够正确保存信息。
- 数据查询（显示）界面应注意显示信息是否正确和完整、是否能正确查询。
- 打印功能要求注意打印出的报表是否正确(包括报表各项信息、数据信息和报表字体等)。
- 对软件的错误处理功能进行测试，即进行错误的操作或输入错误的数据，检查软件对这些情况是否能做出判断并予以提示。
- 特殊情况下要制造极端状态和意外状态，如网络异常中断、电源断电等情况。
- 回归测试的关联性一定要引起充分的注意，避免修改一个错误而引起更多错误出现的现象。
- 妥善保存一切测试过程文档，其意义是不言而喻的，测试的重现性往往要靠测试文档。

18.4 测试计划说明书写作的内容

测试计划说明书是项目经理或开发项目的负责人编写的，并传递给最终用户、系统集成人员、测试人员、软件开发人员、软件管理人员。

- 最终用户用来核实软件开发、测试实施任务和时间人员安排。
- 核实测试需求是否可接受。
- 核实是否使用了适当的测试策略，反映出系统或应用程序按照预定的用途进行应用。

系统集成人员、测试人员、软件开发人员、软件管理人员用来安排工作进度，为整个测试工作指明方向。

《ANSI/IEEE 软件测试文档标准 829-1983》将测试计划定义为：一个叙述了预定的测试活动的范围、途径、资源及进度安排的文档。它确认了测试项、被测特征、测试任务、人员安排，以及任何偶发事件的风险。

软件测试计划是指导测试过程的纲领性文件，包含了产品概述、测试策略、测试方法、测试区域、测试配置、测试周期、测试资源、测试交流、风险分析等内容。

测试计划的目的是：粗略地估计测试大致需要的周期和最终测试报告递交的时间。

测试计划是针对测试中的每个环节的，单元测试、集成测试、系统测试等一般都写测试计划，但编写的重点不同。测试计划为整个测试阶段的管理工作和技术工作提供指南，并确定测试的内容和范围，为评价系统提供依据。

测试计划不是测试阶段时才制定的，测试计划是在各个阶段同步进行相应的测试计划编制，如图 18-2 所示。

<table>
<tr><th>阶段
测试类型</th><th>需求</th><th>设计</th><th>编码</th><th>单元</th><th>集成</th><th>系统</th><th>验收</th></tr>
<tr><td>验收测试</td><td>计划</td><td colspan="5">设计</td><td>执行</td></tr>
<tr><td>系统测试</td><td>计划</td><td colspan="4">设计</td><td>执行</td><td></td></tr>
<tr><td>集成测试</td><td></td><td>计划</td><td colspan="2">设计</td><td>执行</td><td></td><td></td></tr>
<tr><td>单元测试</td><td></td><td></td><td>计划/设计</td><td>执行</td><td></td><td></td><td></td></tr>
</table>

图 18-2　测试计划制定阶段

如图 18-2 所述，各阶段可以同步进行相应的测试计划编制，而测试设计也可以结合在开发过程中实现并行，测试的实施，即执行测试的活动即可连贯在开发之后。值得注意的是：单元测试往往由开发人员承担，因此这部分的阶段划分可能会安排在开发计划，而不是测试计划中。

编写测试计划是一项系统工作，编写者必须对项目了解，对测试工作所接触到的方方面面都要有系统地把握，因此一般情况下是由具有丰富经验的项目测试负责人进行编写，测试计划说明书不能写得冗长、长篇大论、重点不突出，既浪费写作时间，也浪费测试人员的阅读时间。

根据 IEEE 829-1998《软件测试文档》编制标准的建议，测试计划应包含如下 16 项内容：

- 测试计划标识符。
- 介绍。
- 测试项。
- 需要测试的功能。
- 方法（策略）。
- 不需要测试的功能。
- 测试项通过/失败的标准。
- 测试中断和恢复的规定。
- 测试完成所提交的材料。
- 测试任务。
- 环境需求。
- 职责。
- 人员安排与培训需求。
- 进度表。
- 潜在的问题和风险。
- 审批。

制定测试计划时，由于各软件公司的背景不同，测试计划文档也略有差异。测试计划说明书写作的主要内容如下。

1. 引言

引言应包括编写目的、项目背景、定义、测试范围、重点事项、参考资料等。

- 编写目的：阐明编写测试计划的目的并指明读者对象。
- 项目背景：说明项目的来源、委托单位及主管部门。
- 定义：列出测试计划中所涉及到的专业术语和缩写词等。
- 测试范围：测试计划所包含的测试软件、测试的范围和优先级、哪些需要重点测试、哪些无须测试、无法测试或推迟测试。
- 重点事项：列出需要测试的软件的所有主要功能和测试重点，这部分应该能与测试案例的设计相对应并能够互相检查。
- 参考资料：列出有关资料的作者、标题、编号、发表日期、出版单位或资料来源，可包括项目的计划任务书、合同或批文；项目开发计划；项目需求说明书；项目概要设计说明书；项目详细设计说明书；项目用户操作手册；本测试计划中引用的其他资料、采用的软件开发标准或规范等。

2. 测试任务概述

测试任务概述应包括如下内容：

- 测试目标。
- 测试环境，如硬件（列出进行本次测试所需的硬件资源的型号、配置和厂家）和软件（列出进行本次测试所需的软件资源，包括操作系统和支持软件的名称、版本、厂家）。
- 需求概述。
- 条件与限制。
- 测试的方法（策略）描述测试小组用于测试整体和每个阶段的方法，要描述如何公正、客观地开展测试，需要考虑模块、功能、整体、系统、版本、压力、性能、配置和安装等各个因素的影响，尽可能地考虑到细节，越详细越好。

3. 测试计划

测试计划应包括如下内容。

- 测试方案：说明确定测试方法和选取测试用例的原则。
- 测试项目：列出每一项测试的内容、名称、目的和进度。
- 测试准备。
- 测试机构及人员：包括测试机构的名称、负责人和职责；测试人员的人数、经验和专长。测试人员的工作职责明确指出了测试任务和测试人员的工作责任。

有时测试需要定义的任务类型不容易分清，不像程序员所编写的程序那样明确。复杂的任务可能有多个执行者或者由多人共同负责。

4. 测试项目说明

按顺序逐个对测试项目进行说明：

- 测试项目名称及测试内容。

- 测试用例，需要给出输入（是指输入的数据和输入命令）、输出（是指预期的输出数据）、步骤及操作、允许偏差（给出实测结果与预期结果之间允许偏差的范围）。
- 测试进度安排，包括给出进行测试工作的时间安排、测试需要配备哪些人员及如何分工。
- 条件，包括测试对资源的特殊要求，如设备、软件、人员等。
- 测试资料：说明项测试所需要的资料。

5. 评价

评价包括范围和准则。

- 范围：说明所完成的各项测试说明问题的范围及其局限性。
- 准则：说明评价测试结果的准则。

6. 测试数据的记录、整理和分析

说明对本次测试得到数据的记录、整理、分析的方法和存档要求。

7. 测试计划的审核和批准人

测试计划作为质量的重要文档呈现给管理层审核和批准。

18.5 测试大纲写作的内容

测试大纲一般是由一位对整个系统设计熟悉的设计人员编写测试大纲，明确测试的内容和测试通过的准则，设计完整合理的测试用例，以便系统实现后进行全面测试。测试大纲是写测试策略是什么、需要做哪些测试、测试过程如何组织、测试人员包括哪些。测试大纲是测试单位为了获得测试任务在项目招标阶段编制的文件，它是测试单位参与投标时投标书内容的重要组成部分。编制测试大纲的目的是要使建设单位信服，采用本测试单位制定的测试方案，能够圆满实现建设单位的投资目标和建设意图，进而赢得竞争投标的胜利。由此可见，测试大纲的作用是为测试单位的经营目标服务的，起着承接测试任务的作用。

建设单位选择合适的测试单位时，测试大纲用来审查软件测试实施如何展开、进行哪些测试、测试的完整性、测试单位的测试能力。

赢得竞争投标的胜利并签订合同后，测试大纲是项目经理或开发项目的负责人编写测试计划的依据；使系统集成人员、测试人员、软件开发人员、软件管理人员用来安排测试工作和测试内容。测试大纲写作的篇幅较大，要重点突出，测试大纲也是其他说明书写作的基础。

1. 概述

（1）编写目的：测试大纲文档的编写目的在于为某（软件名称）软件测试人员提供详细的测试步骤和测试数据，以保证测试人员对软件测试的正确性和完整性。

（2）术语和缩写词。

（3）参考资料：说明软件测试所需要的资料（需求分析、设计规范等）。

2. 测试环境

（1）硬件：需要列出进行本次测试所需要的硬件资源的型号、配置和厂家。

（2）软件：需要列出进行本次测试所需要的软件资源，包括操作系统和支持软件（不含待测软件）的名称、版本、厂家。

3. 测试阶段技术

测试阶段的技术说明如表 18-2 所示。

表 18-2　测试阶段技术说明

测试阶段技术	是否采用	说明
自动测试技术	是	核心业务流程采用自动测试技术
审评测试	是	对软件产品功能说明文档和设计说明文档进行检查,在需求与设计阶段进行
编写测试用例	是	在产品编码阶段编写测试用例
单元测试	是	由开发人员进行操作
功能测试	是	由开发人员进行操作
集成测试	是	检测模块集成后的系统是否达到需求、对业务流程及数据流的处理是否符合标准、系统对业务流处理是否存在逻辑不严谨及错误、是否存在不合理的标准及要求
性能测试	是	由测试人员进行操作
确认测试	是	在产品发布前，对照 feature list 进行基本需求的确认，确认产品是否正确实现了功能
系统测试	是	包括性能测试、压力测试和回归测试等
验收测试	是	由建设单位、工程实施人员进行操作

4. 测试内容和测试的重点

（1）测试概述：对测试进行一个总体描述。

（2）测试操作步骤的记录：对各测试操作按先后顺序进行编号。具体测试操作步骤的记录如表 18-3 所示。

表 18-3　测试操作步骤的记录

测试名称		标识符	
测试时间		测试人	
操作序号		错误等级	
测试输入	说明输入的具体数据或动作		
预期输出	说明预期的输出或结果		
实际输出	说明实际的输出或结果		
操作序号		错误等级	
测试输入	说明输入的具体数据或动作		
预期输出	说明预期的输出或结果		
实际输出	说明实际的输出或结果		
操作序号		错误等级	
…			

（3）流程测试的要点。

（4）边界值测试的要点。

（5）容错性测试的要点。

（6）异常测试的要点。

（7）安装测试的要点。

（8）易用性测试的要点。

（9）界面测试的要点。

（10）接口测试的要点。

（11）配置测试的要点。

（12）代码会审的要点。

（13）测试环境的要点。

（14）文件传输测试的要点。

（15）数据导入导出测试的要点。

（16）安全性和访问控制测试的要点。

（17）性能测试的要点。

（18）压力测试的要点。

（19）兼容性测试的要点。

（20）升级测试的要点。

（21）功能测试的要点。

（22）单元测试的要点。

（23）集成测试的要点。

（24）系统测试的要点。

（25）回归测试的要点。

（26）验收测试的要点。

（27）运行测试的要点。

（28）文档测试的要点。

（29）测试文档编写。

5. 人员和时间

需要列出一份清单，用于说明在整个测试期间人员的数量、时间、技术水平的要求，以及项目参与人员的职务、姓名、E-mail 和电话，如表 18-4 所示。

表 18-4 项目参与人员

职务	姓名	E-mail	电话
开发工程师			
开发经理			
测试负责人			
测试人员			
…			

6. 进度计划

对进度计划的说明如表 18-5 所示。

表 18-5 进度计划

测试阶段	开始时间	完成时间	测试人员	阶段完成标志
制定测试计划				
需求 Review				
设计 Review				
设计测试用例				
测试开发				
测试环境准备				
测试实施				
功能测试				
集成测试				
性能测试				
系统测试				
验收测试				
文档编写				

7. 测试提交文档

对测试提交文档的说明如表 18-6 所示。

表 18-6 测试提交文档

文档说明	作者	文档位置
测试大纲		
测试计划说明书		
测试需求说明书		
测试任务说明书		
测试用例说明书		
缺陷跟踪报告		
测试分析报告		
程序错误报告		
集成测试报告		

（续表）

文档说明	作者	文档位置
单元测试报告		
测试总结报告		
测试验收报告		
回归测试报告		
功能测试报告		
性能测试报告		
《产品操作手册（后台）》		
《产品操作手册（前台）》		
《产品安装维护手册》		
《产品错误代码说明文档》		

18.6　测试用例写作的内容

测试用例是软件测试的核心，测试用例的设计和编写是软件测试活动中最重要的。设计和编写测试用例的方法有：一般测试用例编写方法、需求测试用例编写方法、接口测试用例编写方法、路径测试的检查用例编写方法、功能测试用例编写方法、健壮性测试-容错能力/恢复能力测试用例编写方法、性能测试用例编写方法、界面测试用例编写方法、信息安全测试用例编写方法、压力测试用例编写方法、可靠性测试用例编写方法、安装/反安装测试用例编写方法等。

1. 一般测试用例编写方法

一般测试用例编写方法的说明如表 18-7 所示。

表 18-7　一般测试用例编写方法

用例编号		测试优先级			
用例摘要					
测试类型					
用例类型					
用例设计者		设计日期		对应需求编号	
对应 UI					
对应 UC					
版本号		对应开发人员			
前置条件					
测试方法					
输入数据					
执行步骤					
预期输出					
实际结果					
测试日期					
结论					

2. 需求测试用例编写方法

需求测试用例编写方法的说明如表 18-8 所示。

表 18-8　需求测试用例编写方法

客户需求列表——需求说明书		开发人员——系统说明书——功能列表		测试人员——功能点测试列表
1	注册功能	1	用户可以自动注册	（对比发现问题）
2		2		
⋮		⋮		
n		n		

3. 接口测试用例编写方法

对接口测试用例编写方法的说明如表 18-9 所示。

表 18-9　接口测试用例编写方法

接口 A 的函数原型		
输入/动作	期望的输出/响应	实际情况
典型值		
边界值		
异常值		
接口 B 的函数原型		
输入/动作	期望的输出/响应	实际情况
典型值		
边界值		
异常值		
…		

4. 路径测试的检查用例编写方法

对路径测试的检查用例编写方法的说明如表 18-10 所示。

表 18-10　路径测试的检查用例编写方法

检查项		结论
数据类型问题	存在不同数据类型的赋值吗	
	变量的数据类型有错误吗	
	存在不同数据类型的比较吗	
变量值问题	变量的初始化或缺省值有错误吗	
	变量发生上溢或下溢吗	
	变量的精度不够吗	
逻辑判断问题	表达式中的优先级有误吗	
	由于精度原因导致比较无效吗	
	逻辑判断结果颠倒吗	

（续表）

检查项		结论
循环问题	错误地修改循环变量吗	
	循环终止条件不正确吗	
	无法正常终止（死循环）吗	
	存在误差累积吗	
内存问题	内存越界吗	
	内存被释放后却继续被使用吗	
	出现野指针吗	
	内存没有被正确地初始化却被使用吗	
	内存泄漏吗	
文件 I/O 问题	文件结束判断不正确吗	
	没有正确地关闭文件吗	
	对不存在的或者错误的文件进行操作吗	
	文件以不正确的方式打开吗	
错误处理问题	忘记进行错误处理吗	
	错误处理程序块一直没有机会被运行吗	
	错误处理程序块本身有毛病吗？如报告的错误与实际错误不一致、处理方式不正确等	
	错误处理程序块是“马后炮”吗？如在被它调用之前软件已经出错	
…	…	

5. 功能测试用例编写方法

对功能测试用例编写方法的说明如表 18-11 所示。

表 18-11　功能测试用例编写方法

功能 A 描述		
用例目的		
前提条件		
输入/动作	期望的输出/响应	实际情况
示例：典型值		
示例：边界值		
示例：异常值		
功能 B 描述		
用例目的		
前提条件		
输入/动作	期望的输出/响应	实际情况
…		

6. 健壮性测试-容错能力/恢复能力测试用例编写方法

对健壮性测试-容错能力/恢复能力测试用例编写方法的说明如表 18-12 所示。

表 18-12 健壮性测试-容错能力/恢复能力测试用例编写方法

异常输入/动作	容错能力/恢复能力	造成的危害、损失
示例：错误的数据类型		
示例：定义域外的值		
示例：错误的操作顺序		
示例：异常中断通信		
示例：异常关闭某个功能		
示例：负荷超出了极限		
…		

7. 性能测试用例编写方法

对性能测试用例编写方法的说明如表 18-13 所示。

表 18-13 性能测试用例编写方法

性能 A 描述		
用例目的		
前提条件		
输入数据	期望的性能（平均值）	实际性能（平均值）
…		
性能 B 描述		
用例目的		
前提条件		
输入数据	期望的性能（平均值）	实际性能（平均值）
…		

8. 界面测试用例编写方法

对界面测试用例编写方法的说明如表 18-14 所示。

表 18-14 界面测试用例编写方法

检查项	测试人员的类别及其评价
窗口切换、移动、改变大小时正常吗	
各种界面元素的文字正确吗（如标题、提示等）	
各种界面元素的状态正确吗（如有效、无效、选中等状态）	
各种界面元素支持键盘操作吗	
各种界面元素支持鼠标操作吗	
对话框中的缺省焦点正确吗	
数据项能正确回显吗	
对于常用的功能，用户能否不必阅读手册就能使用呢	
执行有风险的操作时，有“确认”、“放弃”等提示吗	

（续表）

检查项	测试人员的类别及其评价
操作顺序合理吗	
有联机帮助吗	
各种界面元素的布局合理、美观吗	
各种界面元素的颜色协调吗	
各种界面元素的形状美观吗	
字体美观吗	
图标直观吗	
…	

9. 信息安全测试用例编写方法

对信息安全测试用例编写方法的说明如表 18-15 所示。

表 18-15　信息安全测试用例编写方法

假想目标 A		
前提条件		
非法入侵手段	是否实现目标	代价-利益分析
…		
假想目标 B		
前提条件		
非法入侵手段	是否实现目标	代价-利益分析
…		

10. 压力测试用例编写方法

对压力测试用例编写方法的说明如表 18-16 所示。

表 18-16　压力测试用例编写方法

极限名称 A	例如最大并发用户数量	
前提条件		
输入/动作	输出/响应	是否能正常运行
例如 10 个用户并发操作		
例如 20 个用户并发操作		
…		
极限名称 B		
前提条件		
输入/动作	输出/响应	是否能正常运行
…		

11. 可靠性测试用例编写方法

对可靠性测试用例编写方法的说明如表 18-17 所示。

表 18-17　可靠性测试用例编写方法

任务 A 描述	
连续运行时间	
故障发生的时刻	故障描述
1	
2	
…	
统计分析	
任务 A 无故障运行的平均时间间隔	CPU 小时
任务 A 无故障运行的最小时间间隔	CPU 小时
任务 A 无故障运行的最大时间间隔	CPU 小时
任务 B 描述	
连续运行时间	
故障发生的时刻	故障描述
1	
2	
…	
统计分析	
任务 B 无故障运行的平均时间间隔	CPU 小时
任务 B 无故障运行的最小时间间隔	CPU 小时
任务 B 无故障运行的最大时间间隔	CPU 小时
…	

12. 安装/反安装测试用例编写方法

对安装/反安装测试用例编写方法的说明如表 18-18 所示。

表 18-18　安装/反安装测试用例编写方法

配置说明：		
安装选项	描述是否正常	使用难易程度
全部		
部分		
升级		
其他		
反安装选项	描述是否正常	使用难易程度
全部		
部分		
升级		
其他		

18.7 测试分析报告写作的内容

测试分析报告是测试主要报告之一。测试分析报告是建立在正确的、足够的测试结果的基础之上，不仅要提供必要的测试结果的实际数据，同时要对结果进行分析、对产品质量进行准确的评估。

18.7.1 测试分析报告模板的目录

供参考的测试分析报告模板如图 18-3 所示。

测试分析报告模板目录

1. 概述
    - 1.1 项目简介
    - 1.2 编写目的
    - 1.3 术语定义
    - 1.4 测试环境
    - 1.5 测试人员安排和分工
    - 1.6 参考资料
2. 测试内容
    - 2.1 系统用户使用
    - 2.2 系统功能需求
    - 2.3 系统性能需求
    - 2.4 系统接口需求
    - 2.5 用户界面测试报告
    - 2.6 功能测试报告
    - 2.7 性能测试报告
    - 2.8 接口测试报告
    - 2.9 数据库测试
    - 2.10 安装、卸载测试
3. 测试发现的问题
    - 3.1 功能测试不符合项汇总表
    - 3.2 性能测试不符合项汇总表
    - 3.3 接口测试不符合项汇总表
    - 3.4 软件 Bug 汇总表
4. 测试结果分析
    - 4.1 覆盖分析
    - 4.2 缺陷的统计与分析
5. 测试资源消耗
    - 5.1 测试组织和人员
    - 5.2 测试时间
    - 5.3 资源的总投入
6. 分析与评价
    - 6.1 能力
    - 6.2 缺陷和限制
    - 6.3 评价
7. 测试结论与建议
    - 7.1 测试结论
    - 7.2 建议
8. 程序错误报告写作的内容

图 18-3 供参考的测试分析报告模板

18.7.2 测试分析报告模板的写作内容

1. 概述

（1）项目简介

项目简介需要简要介绍项目的基本情况。

（2）编写目的

编写测试分析报告是为达到以下目的：

- 对系统测试工作进行总结。
- 对测试的各个阶段进行评价，并对测试结果进行分析。
- 为纠正软件缺陷提供依据。
- 指出预期的阅读范围。

（3）术语定义

说明测试分析报告所涉及到的专业术语和缩写词等。

（4）测试环境

列出系统验收测试使用的软硬件环境。如果系统/项目比较大，则用表格方式列出，如表 18-19 所示。

表 18-19　系统验收测试使用的软硬件环境

数据库服务器配置		客户端配置	
内存		CPU	
操作系统		硬盘	
机器网络名		应用软件	
应用服务器配置		局域网地址	
…		…	

（5）测试人员的安排和分工

需要列出本次系统测试实际的人员分工，如表 18-20 所示。

表 18-20　系统测试实际的人员分工

人员	单位	职务	分工
测试负责人			
测试员 1			
测试员 2			
…			
测试员 n			

（6）参考资料

参考资料需要列出如下内容：

- 列出系统测试所参考的资料，需求分析、系统设计、用户手册、本项目的经核准的计划任务书或合同、上级机关的批文。
- 属于本项目的其他已发表的文件。
- 本文件中各处引用的文件、资料，包括所要用到的软件开发标准。
- 列出这些文件的标题、文件编号、发表日期和出版单位，说明能够得到这些文件资料的来源。

2. 测试内容

根据测试计划中编写的测试用例，用表格的形式列出每一项测试的标识符及其测试内容，并指明实际进行的测试工作内容与测试计划中预先设计的内容之间的差别，说明作出这种改变的原因。

（1）系统用户使用

需要列出系统用户使用点，如表 18-21 所示。

表 18-21　系统用户使用点

编号	使用部门	使用岗位	输入内容	输出内容
1				
2				
3				
…				
n				

（2）系统功能需求

需要列出系统功能需求点，如表 18-22 所示。

表 18-22　功能需求点

编号	功能名称	使用部门	功能描述
1			
2			
3			
…			
n			

（3）系统性能需求

需要列出系统性能需求点，如表 18-23 所示。

表 18-23　性能需求点

编号	性能名称	使用部门	性能描述
1			
2			
3			
…			
n			

（4）系统接口需求

需要列出系统接口需求点，如表 18-24 所示。

表 18-24　系统接口需求点

编号	接口名称	接口规范/标准	入口参数	出口参数	传输频率
1					
2					
3					
…					
n					

（5）用户界面测试报告

按照用户界面描述列表内容，设计测试用例（输入/输出）内容，并进行现场测试、记录测试数据、评定测试结果。用户界面测试报告记录如表 18-25 所示。

表 18-25　用户界面测试列表

编号	界面名称	界面描述	输入内容	输出内容	发现问题	测试结果	测试时间	测试人
1								
2								
3								
…								
n								

（6）功能测试报告

按照系统用户的功能需求，设计测试用例（输入/输出）内容，并进行现场测试、记录测试数据、评定测试结果。功能测试报告记录如表 18-26 所示。

表 18-26　功能测试记录表

编号	功能名称	功能描述	输入内容	输出内容	发现问题	测试结果	测试时间	测试人
1								
2								
3								
…								
n								

（7）性能测试报告

按照系统性能需求，设计测试用例（输入/输出）内容，并进行现场测试、记录测试数据、评定测试结果。性能测试报告记录如表 18-27 所示。

表 18-27 性能测试记录表

编号	性能名称	性能描述	输入内容	输出内容	发现问题	测试结果	测试时间	测试人
1								
2								
3								
…								
n								

（8）接口测试报告

按照系统接口需求，进行现场测试、记录测试数据、评定测试结果。接口测试报告记录如表 18-28 所示。

表 18-28 接口测试记录表

编号	接口名称	入口参数	出口参数	传输频率	发现问题	测试结果	测试时间	测试人
1								
2								
3								
…								
n								

（9）数据库测试

需要进行如下测试：

- 数据库操作响应时间。
- 数据库容量。
- 数据库设计检查。
- 数据库连接。

（10）安装、卸载测试

需要进行如下测试：

- 安装正确性和完整性核对。
- 安装、卸载测试兼容性检查点。

3. 测试发现的问题

汇总测试发现的问题，通过汇总的测试结果对软件能力进行全面分析，并提出改进建议（需要标明遗留缺陷、局限性和软件的约束限制等）。

（1）功能测试不符合项汇总表

功能测试不符合项汇总表如表 18-29 所示。

表 18-29　功能测试不符合项汇总表

编号	功能名称	功能描述	输入内容	输出内容	发现问题	测试结果	测试时间	测试人
1						×		
2						×		
3						×		
…						×		
n								

（2）性能测试不符合项汇总表

性能测试不符合项汇总表如表 18-30 所示。

表 18-30　性能测试不符合项汇总表

编号	性能名称	性能描述	输入内容	输出内容	发现问题	测试结果	测试时间	测试人
1						×		
2						×		
3						×		
…						×		
n								

（3）接口测试不符合项汇总表

接口测试不符合项汇总表如表 18-31 所示。

表 18-31　接口测试不符合项汇总表

编号	接口名称	入口参数	出口参数	传输频率	发现问题	测试结果	测试时间	测试人
1						×		
2						×		
3						×		
…								
n								

（4）软件 Bug 汇总表

软件 Bug 汇总表如表 18-32 所示。

表 18-32　软件 Bug 汇总表

	A 类	B 类	C 类	D 类	E 类
Bug 数量					
所占比例（%）					

4. 测试结果分析

（1）覆盖分析

覆盖分析主要包括需求覆盖、测试覆盖，其中需求覆盖如表 18-33 所示。

表 18-33 需求覆盖表

需求/功能（或编号）	测试类型	是否通过	需求覆盖率	备注
Y	P	N/A		

对上表的说明如下。

- Y：表示项。
- P：表示通过。
- N/A：表示不可测试或者用例不适用。
- 需求覆盖率是指经过测试的需求/功能和需求规格说明书中所有需求/功能的比值，通常情况下要达到 100%的目标。需求覆盖率的计算方法是：Y 项/需求总数。

根据测试结果，按编号给出每一测试需求是否通过的结论。

测试覆盖如表 18-34 所示。

表 18-34 测试覆盖

需求/功能（或编号）	用例个数	执行总数	未执行	未/漏测分析和原因	测试覆盖率（%）

其中，测试覆盖率的计算方法是：执行数/用例总数。

（2）缺陷的统计与分析

缺陷统计主要涉及到被测系统的质量，因此，其成为开发人员、质量人员重点关注的部分。

① 缺陷按严重程度汇总

缺陷按严重程度汇总如表 18-35 所示。

表 18-35 缺陷按严重程度汇总

	严重	一般	微小
被测系统缺陷严重程度			

可以给出缺陷的饼状图和柱状图以便直观查看。

② 缺陷按类型汇总

缺陷按类型汇总如表 18-36 所示。

表 18-36　缺陷按类型汇总

	用户界面	一致性	功能	算法	接口	文档	用户界面	其他
被测系统缺陷类型								

可以给出缺陷的饼状图和柱状图以便直观查看。

③ 缺陷按功能分布汇总

缺陷按功能分布汇总如表 18-37 所示。

表 18-37　缺陷按功能分布汇总

	功能 1	功能 2	功能 3	功能 4	功能 5	功能 6	功能 7	功能 n
功能分布								

可以给出缺陷的饼状图和柱状图以便直观查看。

下面通过实例进行介绍。一个被测系统缺陷分布情况如表 18-38 所示。

表 18-38　被测系统缺陷分布情况

	A	B	C	D	E
Bug 数量	2	17	3	0	1
所占比例	9%	74%	13%	0%	4%

缺陷的饼状图如图 18-4 所示。

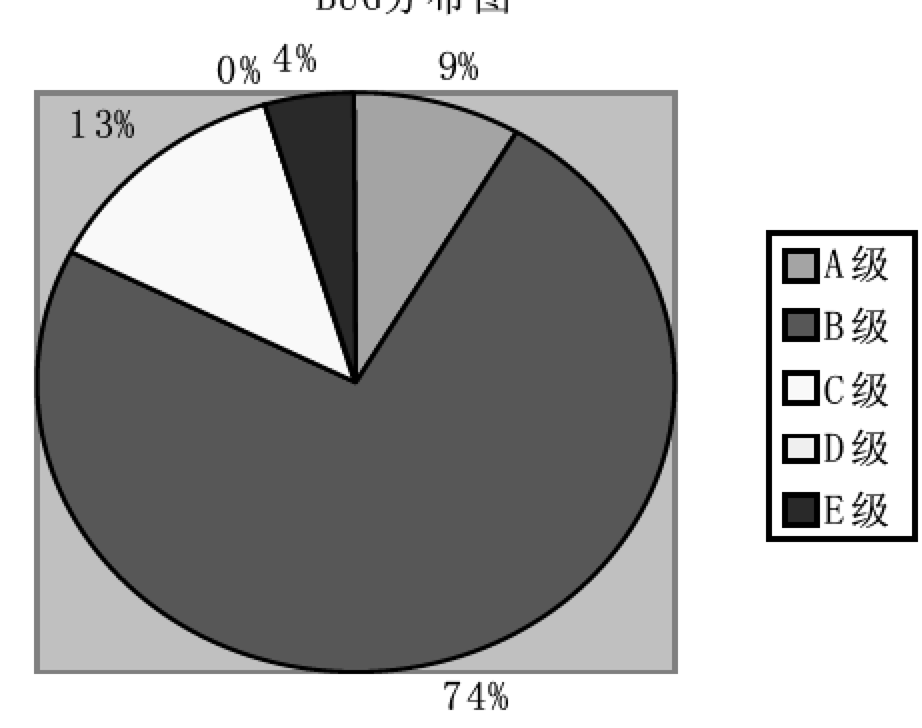


图 18-4　饼状图

缺陷的柱状图如图 18-5 所示。

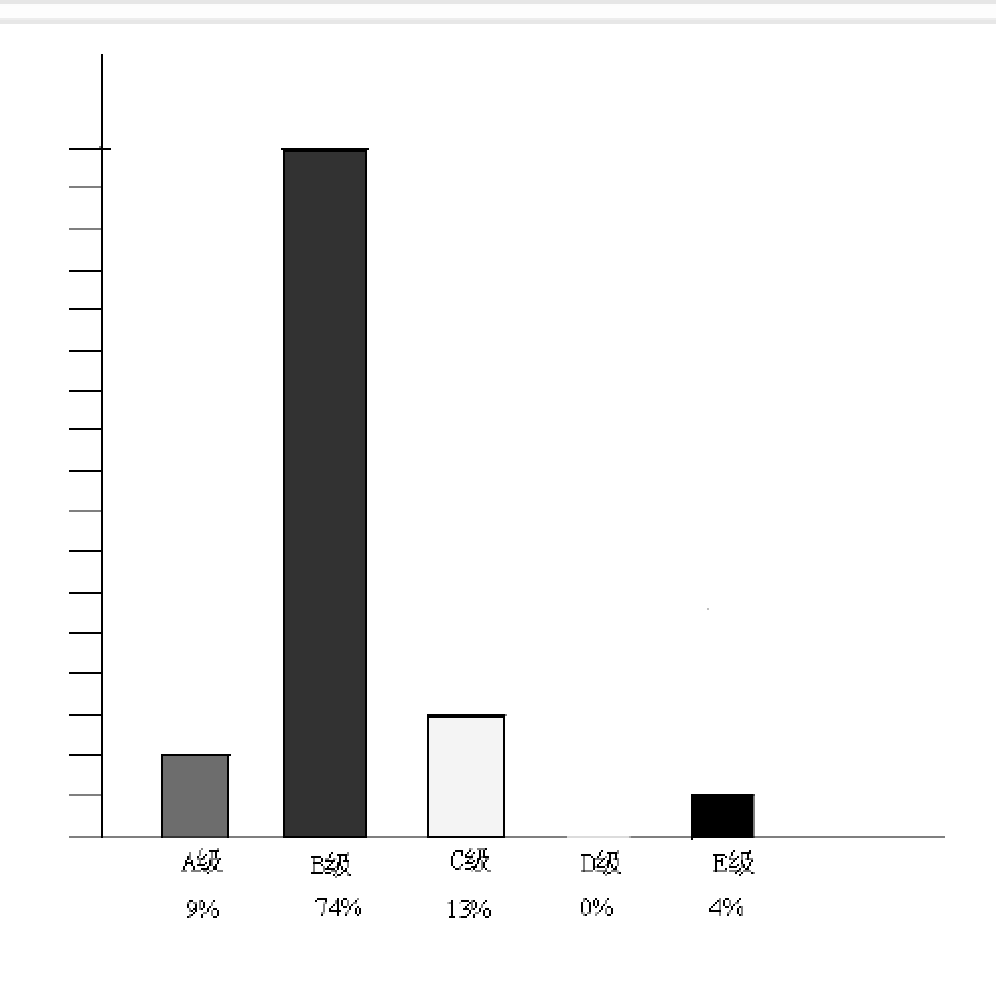


图 18-5 柱状图

5. 测试资源消耗

测试资源消耗用于总结测试工作的资源消耗数据，如工作人员的水平级别数量、机时消耗等，并描述测试资源消耗情况、记录实际数据（测试、项目经理关注的部分）。

（1）测试组织和人员

测试组织和人员主要由如下部分组成：

- 用户工作人员。
- 测试经理（领导人员）。
- 主要测试人员。
- 参与测试人员。

（2）测试时间

测试时间列出测试的跨度和工作量，最好区分测试文档和活动的时间。数据可供过程度量使用。其中总工时的计算方法是：总工作日×8（时）。

测试时间如表 18-39 所示。

表 18-39 测试时间

人员	开始时间	结束时间	合计总工时
用户工作人员			
测试经理			
主要测试人员			
参与测试人员			
总计			

（3）资源的总投入

对于大系统/项目来说，最终要统计资源的总投入，必要时要增加成本一栏，以便管理者清楚

地了解究竟花费了多少人力去完成测试。资源的总投入如表 18-40 所示。

表 18-40 资源的总投入

	人员成本	机时消耗	工具设备	其他费用
资源的总投入				
总计				

6. 分析与评价

（1）能力

陈述经过测试证实了的本软件的能力。

（2）缺陷和限制

陈述经测试证实的软件缺陷和限制，说明每项缺陷和限制对软件性能的影响，并说明全部测得的性能缺陷的累积影响和总影响。

（3）评价

通过对测试结果的分析提出一个对软件能力的全面分析，需要标明遗留缺陷、局限性和软件的约束限制等，说明该项软件的开发是否已达到预定目标，根据测试标准及测试结果，判定软件能否交付使用。

7. 测试结论与建议

（1）测试结论

测试结论如下：

- 测试执行是否充分（可以增加对安全性、可靠性、可维护性和功能性描述）。
- 对测试风险的控制措施和成效。
- 测试目标是否完成。
- 测试是否通过。

（2）建议

可以从以下方面提出建议：

- 对系统存在问题的说明，描述测试所揭露的软件缺陷和不足，以及可能给软件实施和运行带来的影响。
- 可能存在的潜在缺陷和后续工作。
- 对缺陷修改和产品设计的建议。
- 对过程改进方面的建议。

8. 程序错误报告写作的内容

程序错误将会导致程序停止运行，程序错误报告编写的内容如表 18-41 所示。

表 18-41 程序错误报告

系统名称：

测试项目			测试类型	
模块名称			版本	
测试时间			测试批次	
序号	错误等级	错误描述	修改情况	

测试人：

18.8 集成测试报告写作的内容

集成测试是由专门的测试小组来进行的，目的是确保各单元组合在一起后能够按既定意图协作运行。在完成集成测试工作后，测试小组应负责对测试结果进行整理、分析，并形成测试报告。测试报告中要记录实际的测试结果、在测试中发现的问题、解决这些问题的方法以及解决之后再次测试的结果。此外还应提出目前不能解决、还需要管理人员和开发人员注意的一些问题，提供测试评审和最终决策，以提出处理意见。

集成测试报告写作模版如图 18-6 所示。

集成测试报告写作模版
1. 引言
    1.1 编写目的
    1.2 背景
    1.3 定义
    1.4 参考资料
2. 计划集成测试
    2.1 集成测试的制定计划
    2.2 集成测试的具体内容
        2.2.1 功能性测试
        2.2.2 可告性测试
        2.2.3 易用性测试、
        2.2.4 性能测试
        2.2.5 维护性测试
        2.2.6 可移植性测试
        2.2.7 操作性测试
        2.2.8 疲劳性测试
    2.3 集成测试的用例设计
3. 集成测试的结果统计
4. 集成测试的结果评估
5. 集成测试的工作清单
6. 集成测试的审批
7. 集成测试的报告

图 18-6 集成测试报告写作模版

集成测试报告模版写作的主要内容如下。

1. 引言

（1）编写目的：阐述集成测试流程，描述如何进行集成测试活动、如何控制集成测试活动、集成测试活动的流程以及集成测试活动的工作安排。

（2）背景。

（3）定义。

（4）参考资料。

2. 计划集成测试

（1）集成测试的制定计划

在制定集成测试计划时，应考虑如下因素：

- 采用何种系统组装方法来进行组装测试。
- 组装测试过程中连接各个模块的顺序。
- 模块代码编制和测试进度是否与组装测试的顺序一致。
- 测试过程中是否需要专门的硬件设备。
- 在把各个模块连接起来的时候，穿越模块接口的数据是否会丢失。
- 各个子功能组合起来，能否达到预期要求的父功能。
- 一个模块的功能是否会对另一个模块的功能产生不利的影响。
- 全局数据结构是否有问题。
- 单个模块的误差积累起来，是否会放大，从而达到不可接受的程度。

（2）集成测试的具体内容

集成测试的具体内容包括功能性测试、可靠性测试、易用性测试、性能测试、维护性测试、可移植性测试、操作性测试、疲劳性测试等。

① 功能性测试

功能性测试用于检查各个子功能组合起来能否满足设计所要求的功能。

- 一个程序单元或模块的功能是否会对另一个程序单元或模块的功能产生不利影响。
- 根据计算精度的要求，单个程序模块的误差积累起来，是否仍能够达到要求的技术指标。
- 进行程序单元或模块之间的接口测试，即把各个程序单元或模块连接起来时，数据在通过其接口时是否会出现不一致情况、是否会出现数据丢失。
- 全局数据结构的测试，即检查各个程序单元或模块所用到的全局变量是否一致、合理。
- 对程序中可能有的特殊安全性要求进行测试。

② 可靠性测试

根据软件需求和设计中提出的要求，对软件的容错性、易恢复性、错误处理能力进行测试。

③ 易用性测试

根据软件设计中提出的要求，对软件的易理解性、易学性和易操作性进行检查和测试。

④ 性能测试

根据软件需求和设计中提出的要求，进行软件的时间特性、资源特性测试。

⑤ 维护性测试

根据软件需求和设计中提出的要求，对软件的易修改性进行测试。

⑥ 可移植性测试

根据软件需求和设计中提出的要求，对软件在不同操作系统环境下被使用的正确性进行测试。

⑦ 操作性测试

操作性测试主要测试操作是否正确、有无误差，可以分为如下两部分。

- 返回测试：由主界面逐级进入最终界面，按后退键逐级返回，检查返回时屏幕聚焦是否正确。
- 进入测试：由主界面逐级进入最终界面，返回主界面后再次进入，检查是否聚焦正确。

⑧ 疲劳性测试

疲劳性测试主要是通过大容量数据进行测试。

（3）集成测试的用例设计

集成工作版本应分析其类协作与消息序列，从而找出该工作版本的外部接口。由集成工作版本的外部接口确定集成测试用例。测试用例应覆盖工作版本每一外部接口的所有消息流序列。



一个外部接口和测试用例的关系是多对多、一对多，部分集成工作版本的测试需求可映射到系统测试需求，因此对这些集成测试用例可采用重用系统测试用例技术。

3. 集成测试的结果统计

测试用例执行结果统计如表 18-42 所示。

表 18-42 测试用例执行结果统计表

测试项	测试用例号	用例描述	测试结论
功能性测试			
可靠性测试			
易用性测试			
性能测试			
可维护性测试			
可移植性测试			
操作性测试			
返回测试			
进入测试			
疲劳性测试			

4. 集成测试的结果评估

集成测试的结果评估包括以下内容：

- 成功地执行了测试计划中规定的所有集成测试。
- 需要的测试用例和所期望的测试结果。
- 修正了所发现的错误。
- 测试结果通过了专门小组的评审。
- 集成部经理召集本组人员开会讨论，测试结果与测试用例中期望的结果一致，测试通过，否则标明测试未通过。

5. 集成测试的工作清单

集成测试的工作清单如下：

- 软件集成测试计划。
- 集成测试用例。
- 测试过程。
- 测试脚本。
- 测试日志。
- 测试评估摘要。

6. 集成测试的审批

进行审批时，需要填写集成部经理的姓名以及签字日期、技术部经理的姓名及签字日期。

7. 集成测试的报告

集成测试的报告如表 18-43 所示。

表 18-43　集成测试报告

项目名称：集成测试报告		项目编号		
填写人：		测试时间		
测试项目	发现问题	测试结论	测试人	测试负责人
功能性测试				
可靠性测试				
易用性测试				
性能测试				
可维护性测试				
可移植性测试				
操作性测试				
疲劳性测试				
返回测试				
进入测试				
测试结果评估结论： 审批负责人： 年　月　日		审核（项目经理）： 年　月　日		

18.9　单元测试报告写作的内容

单元测试是每一个开发人员都必须去做的事，为了跟踪单元测试的效果，对开发人员进行督促，应编写单元测试报告。

1. 单元测试报告的写作目的

单元测试报告的写作目的在于如下方面：

- 对单元测试结果进行整理和汇总，形成正式的测试文档。
- 为软件单元的评审验收提供依据。
- 纳入软件产品配置管理库。

2. 单元测试报告的写作内容

（1）软件单元描述

简单描述被测试单元或与之相关单元的产品项目名称、所属子系统、单元要完成的功能、需求和设计要求等。

（2）单元结构

画出本单元的组织结构，包括本单元包括的属性、方法、输入/输出等。

（3）单元控制/时序流图

根据本单元的控制结构或操作时序，画出其大概过程。

（4）测试过程

简要描述在本单元的测试过程。

（5）测试

测试主要包括如下几方面。

- 代码审查：代码审查的内容如表 18-44 所示。
- 测试用例：测试用例的执行结果统计如表 18-45 所示。
- 测试特性：指功能测试、性能测试、余量测试、容错性等需要对该子功能进行测试的特性分类。
- 用例描述：用例描述是对该测试用例、测试子功能点的简单描述，如测试打印预览时向下翻页的功能是否实现。
- 测试结论：说明测试是否通过，只需填写“通过”或“不通过”。

表 18-44　代码审查表

Bug	ID	审查人员	审查日期	问题描述

表 18-45　测试用例执行结果统计表

测试项	测试用例号	测试特性	用例描述	测试结论	对应 Bug	ID

（6）提交清单

在单元测试中提交软件 Bug 清单。

（7）质量评估

质量评估是对本测试单元模块的评价，包括功能、性能、余量、人机交互界面、可靠性、可维护性等。

（8）填写单元测试表格

单元测试表格如表 18-46 所示。

表 18-46　单元测试填写表格

填表日期：　　　　　　　　　编号：

<table>
<tr><td colspan="2">开发项目名称</td><td colspan="2"></td><td colspan="2">开发项目编号</td><td></td><td>第一责任人</td><td></td></tr>
<tr><td>单元名称</td><td></td><td>责任人</td><td></td><td colspan="2">单元所属子系统</td><td></td><td>开发周期</td><td></td></tr>
<tr><td colspan="9">代码测试</td></tr>
<tr><td colspan="2">代码测试内容</td><td colspan="2">测试人员</td><td colspan="3">测试结果</td><td colspan="2">备注</td></tr>
<tr><td colspan="2">路径测试</td><td colspan="2"></td><td colspan="3"></td><td colspan="2"></td></tr>
<tr><td colspan="2">循环测试</td><td colspan="2"></td><td colspan="3"></td><td colspan="2"></td></tr>
<tr><td colspan="2">边界测试</td><td colspan="2"></td><td colspan="3"></td><td colspan="2"></td></tr>
<tr><td colspan="2">接口测试</td><td colspan="2"></td><td colspan="3"></td><td colspan="2"></td></tr>
<tr><td colspan="2">界面测试</td><td colspan="2"></td><td colspan="3"></td><td colspan="2"></td></tr>
<tr><td colspan="2">数据确认测试</td><td colspan="2"></td><td colspan="3"></td><td colspan="2"></td></tr>
<tr><td colspan="2">代码走查</td><td colspan="2"></td><td colspan="3"></td><td colspan="2"></td></tr>
<tr><td colspan="9">功能测试</td></tr>
<tr><td>序号</td><td colspan="2">功能名称</td><td>操作方法</td><td>结果</td><td>建议</td><td colspan="2">测试人员</td><td>备注</td></tr>
<tr><td></td><td colspan="2"></td><td></td><td></td><td></td><td colspan="2"></td><td></td></tr>
<tr><td></td><td colspan="2"></td><td></td><td></td><td></td><td colspan="2"></td><td></td></tr>
<tr><td>测试结论</td><td colspan="3"></td><td colspan="2">项目第一责任人</td><td colspan="3"></td></tr>
<tr><td>责任人</td><td colspan="8"></td></tr>
<tr><td>审核</td><td colspan="8"></td></tr>
<tr><td>项目经理</td><td colspan="2"></td><td colspan="2">测试经理</td><td colspan="2"></td><td>总工程师</td><td></td></tr>
</table>

18.10　系统测试总结报告写作的内容

系统测试总结报告是测试阶段最后的文档。测试总结报告的目的是总结测试活动的结果，并根据这些结果对测试进行评价。这种报告是测试人员对测试工作进行的总结。

18.10.1　系统测试总结报告模板的图示

IEEE 829-1998《软件测试文档》编制标准的测试总结报告模板如图 18-7 所示。

IEEE 标准 829－1998 软件测试文档编制标准
测试总结报告模板
目录
1．测试总结报告标识符
2．总结
3．差异
4．综合评估
5．结果总结
    5.1　已解决的意外事件
    5.2　未解决的意外事件
6．评价
7．建议
8．活动总结
9．审批

图 18-7　测试总结报告模板

由于测试总结报告是测试人员对测试活动的总结，这里给出本系统测试总结报告模板，如图 18-8 所示（仅供参考）。

1 简介
    1.1　编写目的
    1.2 项目背景
    1.3 系统简介
    1.4 术语和缩写词
    1.5 测试工具
    1.6 参考资料
2.　测试环境与配置
3.　测试方法
4.　测试总结
    4．1 测试时间、地点、人员
    4．2　测试范围
    4．3 工作组织
    4．4 测试分析
    4．5 残留缺陷与未解决问题
    4．6 测试资源消耗情况
    4．7 测试结论
    4．8 测试文档
5.　建议
6.　附件
    附件1 测试用例清单
    附件2 缺陷清单

图 18-8　系统测试总结报告模板

18.10.2 系统测试总结报告模板的写作要点

1. 简介

简介包括编写目的、项目背景、系统简介、术语和缩写词、测试工具、参考资料等。

2. 测试环境与配置

简要介绍测试环境及其配置。



如果系统/项目比较大，则可利用表格的方式列出。

3. 测试方法

简要介绍测试中采用的方法。



测试方法可以写上测试的重点和采用的测试模式，这样可以一目了然地知道是否遗漏了重要的测试点和关键块。

当使用到测试工具和相关工具时，需要进行说明，如注明是自产还是厂商、版本号是多少，在测试报告发布后要避免大多工具的版权问题。

4. 测试总结

（1）测试时间、地点、人员

需要注明的内容如下：

- 本次测试从 x 年 x 月 x 日至 x 年 x 月 x 日，以及工作地点等。
- 本次测试参与人员，如张三、李四、王五。
- 本次测试张三参与了整个测试期的工作，李四参加了 x 月 x 日至 x 月 x 日的工作，王五参加了 x 月 x 日至 x 月 x 日的工作等。
- 本次测试功能点 x 个，执行 x 个测试用例，测试共发现 x 个 Bug，其中严重级别的 Bug 有 x 个，无效 Bug 有 x 个。

（2）测试范围

本次测试对象为某部电子政务系统，进行了边界值测试、容错性测试、异常测试、安装测试、易用性测试、界面测试、接口测试、配置测试、代码会审、文件传输测试、数据导入导出测试、安全性和访问控制测试、性能测试、压力测试、兼容性测试、升级测试、功能测试、单元测试、集成测试、系统测试、回归测试、验收测试、运行测试、文档测试和测试文档编写。

（3）工作组织

现举例如下：

- x 与 x 进行了任务分析，并编制了《测试大纲》、《测试计划》、《测试需求》。
- x 编制了《测试用例》，独立测试了 x 模块。
- x 进行了 x 测试。
- …

（4）测试分析

需要注明：系统进行了细致地测试，对发现的问题已提交并修改，进行了回归测试。

① 测试统计

对如下内容进行了统计：

- 对测试用例执行结果进行了统计。
- 对系统用户使用点测试结果进行了统计。
- 对系统功能需求测试结果进行了统计。
- 对系统性能需求测试结果进行了统计。
- 对用户界面测试结果进行了统计。
- 对系统接口测试结果进行了统计。
- 对功能测试结果进行了统计。
- 对数据库测试结果进行了统计。
- …

② 测试发现的问题汇总

对如下问题进行了汇总：

- 功能测试不符合项汇总。
- 性能测试不符合项汇总。
- 接口测试不符合项汇总。
- 软件 Bug 汇总。

③ 测试结果分析

测试结果分析包括覆盖分析、测试覆盖分析、缺陷的统计与分析。缺陷统计主要涉及到被测系统的质量，因此，这部分是开发人员、质量人员重点关注的部分。

（5）残留缺陷与未解决问题

对残留缺陷与未解决问题的看法，也就是这些问题如果发现了会造成什么样的影响。
对残留缺陷的说明如下。

- 编号：Bug 号。
- 缺陷概要：该缺陷描述的事实。
- 原因分析：如何引起缺陷、缺陷的后果，并描述造成软件局限性和其他限制性的原因。
- 预防和改进措施：弥补手段和长期策略。

对未解决问题的说明如下：

- 功能/测试类型。
- 测试结果：与预期结果的偏差。
- 缺陷：具体描述。

（6）测试资源消耗情况

总结测试工作的资源消耗数据，如工作人员的水平、级别、数量，以及测试时间和资源的总投入等。

（7）测试结论

测试结论需要从功能性、易用性、可靠性、兼容性、安全性等多方面进行阐述，如系统实现了如下功能性、系统实现了如下易用性等。

（8）测试文档

关于测试文档的内容前面已经进行了介绍，这里不再赘述。

5. 建议

可以从以下方面提出建议：

- 对系统存在问题的说明，描述测试所揭露的软件缺陷和不足，以及可能给软件实施和运行带来的影响。
- 可能存在的潜在缺陷和后续工作。
- 对缺陷修改和产品设计的建议。
- 对过程改进方面的建议。

6. 附件

最后，需要添加如下附件。

- 附件 1：测试用例清单。
- 附件 2：缺陷清单。



本小节内容仅供参考。

18.11 验收测试报告写作的内容

验收测试是依据软件开发商和用户之间的合同、软件需求说明书以及相关行业标准、国家标准、法律法规等对软件的功能、性能、可靠性、易用性、可维护性、可移植性等特性进行严格的测试，验证软件的功能、性能及其他特性是否与业务需求一致。

验收测试报告模版如图 18-9 所示。

验收测试报告模版

1. 概述
   1．1 验收测试目的
   1．2 项目基本情况
   1．3 验收测试范围
2. 验收测试组织方案
   2．1 验收测试时间
   2．2 测试地点
   2．3 验收测试环境
      2．3．1 硬件
      2．3．2 软件
      2．3．3 网络
      2．3．4 测试工具
   2．4 人员安排
3. 项目进度审核
   3．1 项目实施进度情况
   3．2 项目合同变更情况
   3．3 项目需求变更情况
   3．4 项目投资结算情况
4. 验收测试计划
   4．1 验收测试原则
   4．2 验收测试方式
   4．3 验收测试内容
5 项目验收测试情况汇总
   5．1 项目验收测试情况汇总
   5．2 项目验收附件明细
6. 项目验收测试结论
   6．1 开发单位结论
   6．2 建设单位结论
7. 验收测试结果汇总
8. 附件

图 18-9　验收测试报告模版

1. 概述

概述用于描述该验收测试的目的、项目基本情况和验收测试范围。

（1）验收测试目的

验收测试的任务是验证该软件功能、性能及其他特性是否与业务需求一致。此处必须对系统目前状况进行简略描述，并指明通过什么样的测试以达到什么较为具体的目的、预期结果是什么等。

（2）项目基本情况

项目基本情况如表 18-47 所示。

表 18-47　项目基本情况

项目名称	
项目合同甲方	
项目合同乙方	
项目合同编号	
项目开工时间	
项目竣工时间	
项目验收日期	

（3）验收测试范围

根据系统需求说明书、功能说明书和测试大纲所描述的各项功能进行测试。

2. 验收测试组织方案

（1）验收测试时间

描述本次验收测试的进度计划和具体时间安排。

（2）测试地点

描述本次验收测试的地点。

（3）验收测试环境

描述测试环境时需要包括对硬件、软件、网络、测试工具的说明。

- 硬件（主要包括主机、打印机、终端）：说明所需设备的机型要求以及内存、CPU、硬盘大小的最低要求。
- 软件（操作系统、数据库、工具程序）：详细说明每台设备上部署的自开发的、第三方软件的名称和版本号，以便系统管理员按照此说明计划分配测试资源。
- 网络：网络拓扑结构图、网络设备、路由器、交换机、集线器、电话线等。
- 测试工具。

（4）人员安排

说明完成此次测试的人员组成、任务以及各工作小组的职责，如领导小组、工作小组（开发部门、需求部门、质量检部门、业务部门）、项目小组等。

3. 项目进度审核

（1）项目实施进度情况

对项目实施进度情况的说明如表 18-48 所示。

表 18-48　项目实施进度情况

序号	阶段名称	起止时间	交付物列表	备注
1				
2				
3				
4				
5				
6				

（2）项目合同变更情况

项目合同变更情况用于记录合同变更情况。

（3）项目需求变更情况

项目需求变更情况用于记录需求变更情况。

（4）项目投资结算情况

对项目投资结算情况的说明如表 18-49 所示。

表 18-49 项目投资结算情况

序号	款项	金额（万元）	备注
1			
2			
3			
4			
5			
合计			

4. 验收测试计划

（1）验收测试原则

验收测试的原则如下：

- 审查提供验收的各类文档的正确性、完整性和统一性，审查文档是否齐全、合理。
- 审查项目功能是否达到了合同规定的要求。
- 审查项目有关服务指标是否达到了合同的要求。
- 审查项目投资以及实施进度的情况。
- 对项目的技术水平做出评价，并得出项目的验收结论。

（2）验收测试方式

记录项目验收的组织方式和参与验收工作的人员情况如表 18-50 所示。

表 18-50 记录项目验收的组织方式

序号	验收方式	验收人员	所属单位	相关职责
1				
2				
3				
4				
5				
6				

（3）验收测试内容

验收测试的要点如下：

- 流程测试。
- 边界值测试。

- 容错性测试。
- 异常测试。
- 安装测试。
- 易用性测试。
- 界面测试。
- 接口测试。
- 配置测试。
- 环境测试。
- 文件传输测试。
- 数据导入导出测试。
- 安全性和访问控制测试。
- 性能测试。
- 压力测试。
- 兼容性测试。
- 升级测试。
- 功能测试。
- 单元测试。
- 集成测试。
- 系统测试。
- 回归测试。
- 运行测试。
- 文档测试。

5. 项目验收测试情况汇总

项目验收测试情况汇总如表 18-51 所示。

表 18-51　项目验收情况汇总

验收项	验收意见		备注
	通过	不通过	
总体意见： 项目验收组长（签字）： 日期：			

项目验收附件明细包括如下内容：

- 软件平台验收单（见附件一）。
- 功能模块验收单（见附件二）。
- 项目文档验收单（见附件三）。
- 硬件设备验收单（见附件四）。

6. 项目验收测试结论

需要给出结论，如哪些产品通过验收、哪些产品没有通过验收。

开发单位结论如图 18-10 所示。

开发单位结论   开发单位（签章） 日期：

图 18-10　开发单位结论

建设单位结论如图 18-11 所示。

建设单位结论   建设单位（签章） 日期：

图 18-11　建设单位结论

7. 验收测试结果汇总

验收测试结果汇总如表 18-52 所示。

表 18-52　验收结果汇总

验收产品名称	计划交付时间	实际交付时间	通过测试/评审的时间	测试/评审意见	项目组意见
⋮					

8. 附件

附件一（软件平台验收单）如表 18-53 所示。

表 18-53 软件平台验收单

验收人				
验收时间				
序号	软件类型	软件名称	验收结果	备注

附件二（功能模块验收单）如表 18-54 所示。

表 18-54 功能模块验收单

验收人				
验收时间				
序号	功能模块	验收内容	合同要求	验收结果

附件三（项目文档验收单）如表 18-55 所示。

表 18-55 项目文档验收单

验收人				
验收时间				
序号	文档名称	用途	验收结果	备注

附件四（硬件设备验收单）如表 18-56 所示。

表 18-56　硬件设备验收单

验收人						
验收时间						
序号	硬件名称	基本用途	型号	配置情况	验收结果	备注

第19章　软件的其他测试技术

软件的其他测试技术不是一个基本过程测试技术，而是一个辅助的测试技术，本章重点讨论以下内容：

- 可用性测试。
- 安全性测试。
- 强度测试或压力测试。
- 确认测试。
- 容错性测试。
- 软件回归测试技术。
- 易用性测试。

19.1　可用性测试

1. 可用性的概念

可用性是产品的一个基本的自然属性，是最终用户使用产品的可用程度。可用性是在产品和用户的相互作用中体现出来的。可用性的基本评价指标是效率、满意和安全（容错、无错）。

2. 可用性的测试方法

在可用性测试中，可对同一测试内容在同时采用多指标的测试，也可对同一测试内容在不同时间采用多指标的测试。

19.2　安全性测试

安全性是软件程序在其设计、运行环境中，不会引起诱发对人员或设备的危害。软件安全性一般分为两个层次，即应用程序级别的安全性和系统级别的安全性。

安全性测试包括以下方式：

- 想方设法截取或破译口令。
- 专门开发软件来破坏系统的保护机制。
- 故意导致系统失败。
- 试图通过浏览保密数据获得所需信息等。

1. 应用程序级别的安全性测试方法

对数据或业务功能进行访问时，在预期的安全性情况下，操作者只能访问应用程序的特定功

能和有限的数据，也只能访问其所属用户类型已被授权访问的那些功能或数据。

对于不同权限的用户类型，创建各种用户类型并利用各种用户类型所特有的事务来核实其权限，最后修改用户类型，并为相同的用户重新运行测试。

对测试结果的安全性进行分析，包括分析所有测试用例是否通过；测试代码是否按照要求分析，并达到相应的测试覆盖率；对测试结果进行分析，以验证所有的安全性需求是否得到了满足。

2. 系统级别的安全性测试策略和方法

只有具备系统访问权限的用户才能访问应用程序，而且只能通过相应的网关来访问，包括对系统的登录或远程访问。

只有具备系统和应用程序访问权限的操作者才能访问系统和应用程序。

19.3 强度测试或压力测试

强度或压力测试是在一定约束条件下测试系统所能承受的并发用户量、运行时间、数据量，以确定系统所能承受的最大负载压力，以检查程序对异常情况的抵抗能力，找出性能瓶颈。异常情况主要指那些峰值、极限值、大量数据的长时间处理等，负载压力测试是性能测试的重要组成部分，压力测试主要有如下方式：

- 连接或模拟了最大（实际或实际允许）数量的客户机。
- 所有客户机在长时间内执行相同的、性能可能最不稳定的重要业务功能。
- 已达到最大的数据库大小，而且同时执行多个查询或报表事务。
- 当中断的正常频率为每秒 1 ~ 2 个时，运行每秒产生 10 个中断的测试用例。
- 运行可能导致操作系统崩溃或大量数据对磁盘进行存取操作的测试用例等。

压力测试可以分为稳定性测试和破坏性测试。

- 稳定性压力测试：在选定的压力值下，持续运行 24 小时以上的测试。通过压力测试，可以考察各项性能指标是否在指定范围内，有无内存泄漏、有无功能性故障等。
- 破坏性压力测试：在压力稳定性测试中可能会出现一些问题，如系统性能明显，但很难暴露出真实的原因。通过破坏性不断加压的手段，往往能快速造成系统的崩溃或让问题明显地暴露出来。

在压力测试中，会给程序加上一些跟踪机制（如 LOG、日志等），然后查看监视系统、服务器等性能的日志文件，找出问题出现的关键时间或检查测试运行参数，通过分析问题或参数，从而有目的地调整测试策略或测试环境，使压力测试结果真实地反映出软件的性能。

19.4 确认测试

确认测试用于检验所开发的软件是否按照软件需求规格说明书中确定的软件功能、性能、约束及限制等技术条件、要求进行工作。

确认测试内容主要包括功能和性能两部分。

1. 功能测试

功能测试主要考察软件对功能需求完成的情况，应该设计测试用例使需求规定的每一个软件功能得到执行和确认。

- 按照系统给出的功能列表，逐一设计测试案例。
- 对于需要资料合法性和资料边界值检查的功能，增加相应的测试案例。
- 运行测试案例。
- 检查测试结果是否符合业务逻辑。
- 评审功能测试结果。

2. 性能测试

性能测试用于检验软件是否达到需求规格说明中规定的各类性能指标，并满足一些与性能相关的约束和限制条件。

- 测试软件在获得定量结果时程序计算的精确性。
- 测试在有速度要求时完成功能的时间。
- 测试软件完成功能时所处理的数据量。
- 测试软件各部分工作的协调性，如高速操作、低速操作的协调性。
- 测试软件/硬件中因素是否限制了产品的性能。
- 测试产品的负载潜力及程序运行时占用的空间。

19.5　容错性测试

容错测试是一种对抗性的测试过程。容错性测试主要包括以下两个方面的测试。

- 输入异常数据或进行异常操作，以检验系统的保护性。如果系统的容错性好的话，系统只给出提示或内部消化掉，而不会导致系统出错甚至崩溃。
- 灾难恢复性测试。通过各种手段，让软件强制性地发生故障，然后验证系统已保存的用户数据是否丢失、系统和数据是否能很快恢复。

19.6　回归测试技术

回归测试（Regression Testing）作为软件生命周期的一个组成部分，在软件开发的各个阶段都可能需要进行多次回归测试，回归测试在整个软件测试过程中占有很大的工作量比重。

1. 回归测试的定义

在软件生命周期中的任何一个阶段，只要软件发生修改变化时，就必须重新测试现有的功能，以便确定修改是否达到了预期的目的，修改有可能导致软件未被修改的部分产生新的问题，使本来工作正常的功能产生错误；同样，在有新代码加入软件时，除了新加入的代码中有可能含有错误外，新代码还有可能对原有的代码带来影响，就必须重新测试软件的功能，以便确定修改是否达到了预期的目的，检查修改是否损害了原有的正常功能，增添新的测试用例和原有的测试用例对软件再测试。

回归测试不是一个特定的测试级别，只要对软件代码有修改，不论是修改错误还是增加新的功能或是提高性能，原则上都要进行回归测试，以保证对代码修改的正确性，且不会对其余部分带来负面影响。回归测试可以通过重新执行所有的测试用例的一个子集进行，回归测试集包括三种类型的测试用例：

- 能够测试软件的所有功能的代表性测试用例。
- 专门针对可能会被修改影响的软件功能的附加测试。
- 针对修改过的软件成分的测试。

回归测试可以有选择地重复执行集成和系统测试的测试用例，回归测试变动比较小，同时测试所基于的实际硬件环境相对比较稳定。但回归测试要频繁地重复运行，需要的工作量很大，所以，回归测试最值得自动化。回归测试以非常高效的方式进行，软件开发的各个阶段都会进行多次回归测试。

2. 回归测试的方法

回归测试常用的方法如下。

（1）再测试全部用例

把测试用例库中的全部测试用例组成回归测试包进行测试，这是一种最安全的方法，再测试全部用例具有最低的遗漏回归错误的风险，但测试时间、人员、设备和经费成本最高。全部再测试几乎可以应用到任何情况下，基本上不需要进行分析和重新开发，但是，随着开发工作的进展，测试用例不断增多，重复原先所有的测试将带来很大的工作量，往往超出了预算和进度。

（2）基于风险选择测试

从测试用例库中选择测试用例进行回归测试。这种方法有一定的风险。选择测试用例首先要选择运行最重要的、关键的和可疑的测试，而跳过那些非关键的、优先级别低的或者高稳定的测试用例。

（3）基于操作选择测试

如果测试用例库的测试用例是基于软件操作开发的，回归测试可以优先选择基于操作的测试用例，针对最重要或最频繁使用功能的测试用例，这种方法可以在一个给定的预算下最有效地提高系统可靠性，但实施起来有一定的难度。

（4）仅测试修改的部分

当测试者对修改的局部化有足够的信心时，可以仅测试修改的部分，但要分析修改情况和修改的影响，回归测试尽可能覆盖受到影响的部分，但实施起来有一定的风险。

3. 回归测试的测试用例库维护

项目的测试组在实施测试的过程中会将所开发的测试用例保存到“测试用例库”中，并对其进行维护和管理。回归测试“测试用例库”的维护主要包括如下几个方面：

- 删除过时的测试用例。
- 改进不受控制的测试用例。
- 删除冗余的测试用例。
- 增添新的测试用例。

4. 回归测试的基本过程

回归测试的基本过程如下：

01 识别出软件中被修改的部分。

02 从测试用例库中，排除所有不再使用的测试用例，确定那些对新的软件版本依然有效的测试用例，其结果是建立一个新的基线测试用例库。

03 依据一定的策略从新的基线测试用例库中选择测试用例测试被修改的软件。

04 如果必要，可生成新的测试用例集，用于测试新的基线测试用例库无法充分测试的软件部分。

05 用测试用例集执行修改后的软件。

第 02 和第 03 步测试验证修改是否破坏了现有的功能，第 04 和第 05 步测试验证修改工作本身。

19.7 易用性测试

易用性（Useability）是交互的适应性、功能性和有效性的集中体现，是指在指定条件下使用时，软件产品被理解、学习、使用和吸引用户的能力。

易用性一般分为两个层次，即用户界面的易用性和操作系统的易用性。

1. 用户界面测试

用于与软件交互的方式称为用户界面或 UI，易用性包括如下方面的测试。

（1）符合标准和规范

用户界面要素要符合软件现行的标准和规范。

（2）直观

用户界面是否洁净、不拥挤；布局是否合理；是否有多余功能。

（3）一致

如果软件或者平台有一个标准，就要遵守它。如果没有，就要注意软件的特性，确保相似的操作以相似的方式进行。

（4）舒适

软件使用起来应该舒适，不能给用户工作制造障碍和困难。

（5）实用

是否实用是优秀用户界面的最后一个要素。

2. 操作系统的易用性

操作系统有内置的支持，如 Windows 系统提供了粘滞键、筛选键、切换键、声音卫士、声音显示、高对比度、鼠标键、串行键。

第20章　软件测试管理

软件测试管理是以测试项目为管理对象，通过一个临时性的专门的测试组织发挥项目团队的作用，运用专门的软件测试知识、技能、工具和方法，对测试项目进行计划、组织、执行和控制，并在时间成本、软件测试质量等方面进行分析和管理活动。测试管理贯穿整个测试项目的生命周期，强调以人为本对测试项目的全过程进行管理。测试管理是一个很重要的环节，对测试工作相当重要。

本章重点讨论以下内容：

- 测试管理概述。
- 测试项目管理。
- 测试过程管理。
- 软件测试的组织和人员管理。
- 测试配置管理。
- 软件缺陷管理。
- 变更请求管理。
- 测试项目的进度管理。
- 软件测试风险管理。
- 软件测试的成本管理。

20.1　测试管理概述

测试管理就是以测试为管理对象，通过一个临时性的专门的测试组织，利用有限的人力和财力等资源，在指定的环境和要求下，运用专门的软件测试知识、技能、工具和方法，对测试进行计划、组织、执行和控制，并在时间成本、软件测试质量等方面进行分析和管理的活动。

1．测试管理的目的

通过对产品的整个测试流程进行控制和管理，从而提高企业软件测试的管理水平；灌输和强化企业的管理理念；确保开发产品的质量；进一步提高企业的市场竞争能力。

2. 测试管理的特征

测试管理的特征如下：

- 系统工程的思想贯穿测试项目管理的全过程。
- 测试项目管理的组织有一定的特殊性。
- 测试项目管理的要点是创造和保持一个使测试工作顺利进行的环境，使置身于这个环境中的人员能在集体中协调工作以完成预定的目标。

- 测试项目管理的方法、工具和技术手段具有先进性。

3. 测试管理的要素

测试管理具有三个要素，即成本、进度和质量。

4. 测试管理的原则

测试管理具有如下原则：

- 始终能够把质量放在第一位。
- 可靠的需求。
- 尽量留出足够的时间。
- 足够重视测试计划。
- 要适当地引入测试自动化或测试工具。
- 建立独立的测试环境。

5. 测试管理中的 PDCA

P 是指测试计划；D 是指测试案例及测试步骤的设计；C 是指测试实施和错误跟踪；A 是指测试总结报告。

6. 测试管理受的环境影响

测试管理受的环境影响如下：

- 项目组内的环境。
- 项目所处的组织环境。
- 整个开发流程所控制的全局环境。

这三个环境要素直接关系到软件项目的可控性。

20.2 测试项目管理

测试项目是利用有限的人力和财力等资源，在指定的环境和要求下，对特定软件完成特定测试目标的阶段性任务。测试项目要重视测试的策略以提高效率，随时跟踪项目尽量确保项目按计划执行，但更重要的是“质量”。

测试项目应满足一定质量、数量、成本、进度和技术指标等要求。

测试项目一般具有如下特性。

- 项目的独特性。
- 项目的组织性。
- 测试项目的生命期。
- 测试项目的资源消耗特性。
- 测试项目的目标冲突性。
- 具有智力密集、劳动密集的特点。
- 测试项目结果的不确定因素。

测试项目管理具有以下基本特征。

- 系统工程的思想贯穿测试项目管理的全过程。
- 测试项目管理的组织有一定的特殊性。
- 测试项目管理的要点是创造和保持一个使测试工作顺利进行的环境，使置身于这个环境中的人员能在集体中协调工作以完成预定的目标。
- 测试项目管理的方法、工具和技术手段具有先进性。

测试项目管理具有以下基本原则：

- 始终能够把质量放在第一位。
- 可靠的需求。
- 尽量留出足够的时间。
- 足够重视测试计划。
- 要适当地引入测试自动化或测试工具。
- 建立独立的测试环境。
- 建立测试管理方法。

测试项目对管理者有如下 18 点要求：

- 在一个项目中管理者要了解自己的知识面是否与该项目匹配，若不匹配则需要提前做好准备。
- 在一个项目中管理者也要了解测试人员的能力与该项目的要求是否匹配。
- 在一个项目中管理者不要和测试人员争功，上级对管理者的考察永远是团队和项目，帮助测试人员成长和保证项目质量是管理者的责任。
- 在一个项目中管理者的懒惰将会对测试人员和项目造成极坏的影响。
- 在一个项目中管理者要多与开发和产品负责人讨论并了解变化，因为规范不能保证测试的输入没有遗漏。
- 在一个项目中管理者要多参与测试方案、测试用例、测试方法、测试工具、测试过程、测试结果的评审与讨论，弥补测试人员或者管理者考虑不周全的问题。
- 在一个项目中管理者要多考虑测试效率和测试效果的问题，这样可以不断启用新的测试方法和测试流程来提高效率、保证测试效果。
- 在一个项目中管理者要进行阶段小结，这样可以弥补一些测试不足的地方，并很好地规划下一个阶段的计划；测试计划不是一成不变的，必须定期调整。
- 在一个项目中涉及到变更时，要再次评审测试方案、测试用例、测试方法、测试工具，若频繁变更，则更要把握好节奏。
- 在一个项目中管理者要非常重视组件/模块的接口测试、集成测试，不仅表现在方案、用例上，同时也表现在测试时间的安排和人的协调管理上。
- 在一个项目中管理者要非常重视测试人员直接参与技术讨论会议的重要性，既树立测试人员与开发人员沟通的信心，又加深了测试人员对项目的了解情况，对未来的工作开展非常有利。
- 在一个项目中管理者对于还没有掌握沟通技巧或者对管理者没有信心的测试人员，带着测试人员一起和开发或者产品进行沟通，或者鼓励测试人员去沟通，了解测试人员沟通的效

果并指出下次沟通的注意事项。

- 管理者要全面控制和管理测试项目，通过跟踪测试任务、查看测试报告、分析测试结果，实时掌握详细的测试进度。
- 管理者要通过使用完整集中的测试知识库，提高产品的测试质量和管理标准。
- 简化的数据输入形式，可定义的测试界面，以及自动化管理流程，帮助团队有效提高工作效率。
- 管理者要对测试案例、测试数据和测试结果在内的详细历史记录核查，保证了测试工作的可追溯性和可核查性。
- 管理者要全面地测试覆盖管理，创建、管理、分析测试范围，从中心知识库中调用原有的测试范围，以此提高管理者的工作效率、使管理者的管理流程更加标准化。
- 管理者要有高度可视化的测试计划向导，安排测试时间、分配测试任务、调整测试流程。
- 管理者要有质量报表，帮助管理者分析测试趋势、掌握工作进展、总结测试缺陷。

测试经理或测试主管是测试项目成败的关键人物，是对测试项目的成败负主要责任，如果测试经理或测试主管重视并充分发挥测试经理的作用，和项目经理一起制订项目测试大纲，让项目经理在开始阶段更多了解测试的质量需求、结构设计、运行环境。

测试项目对测试经理或测试主管有如下 16 点要求：

- 设置软件测试环境，安装必要的软件工具。
- 运行软件，发现和报告软件缺陷或错误，尤其需要快速定位软件中的严重错误。
- 对软件整体质量提出评估。
- 确认软件达到某种具体标准。
- 以最低的成本、最短的时间，完成高质量的测试任务。
- 在项目开发过程中，随着项目进展，项目经理和测试经理要实时沟通。
- 项目经理需要非常了解项目进度，进行工作任务细化、具体计划和安排项目成员工作任务等工作、对突发事件项目经理需要能及时合理地进行协调、测试经理能准确地给出进展状态和项目的缺陷状态。
- 测试经理不仅需要注重项目质量，同时应注意项目工作效率的不断提高。
- 项目经理对软件开发具有丰富的经验，了解软件开发的普遍流程，了解各个阶段所需完成的工作，特别是项目测试工作需要的时间，这是安排好项目组成员工作的前提。
- 在项目正式开展前，经理准备项目计划文档，在项目计划中包含了项目进度时间表，给出各个阶段和各个子阶段的起始结束日期。对各个阶段和各个子阶段的详细工作安排和各项工作责任人只能在项目开展工程中根据项目实际情况进行安排，一般是在每周项目组例会上进行本周详细工作安排。
- 在项目组例会上的工作安排一般只限于本周甚至是过后的二、三天，一般不会太长，对长时间工作的安排容易失去精确并且不易控制。
- 项目组例会一般一周一次（时间不能太长），但必要时也可在中途召开项目会议进行工作安排。
- 一定重视每周测试结果报告。
- 善于鼓励发挥员工的潜能，经理需要会赞扬很好地完成了工作的组员。
- 要采取主动积极的工作态度和利益相关者去沟通，强化软件测试工作。

- 采用配置管理思想，辅之以先进的配置管理工具，可以帮助用户在内部建立完善的知识管理体系。

20.3 测试过程管理

软件测试和软件开发一样，都有一个过程，测试过程的管理显得尤为重要，过程管理已成为测试成功的重要保证。经过多年努力，测试专家提出了许多测试过程模型，包括 V 模型、W 模型、H 模型等。这些模型定义了测试活动的流程和方法，为测试管理工作提供了指导，但这些模型各有长短，并没有哪种模型能够完全适合于所有的测试项目，在实际测试中应该吸取各模型的长处，提高测试管理水平、测试效率、降低测试成本。

V 模型、W 模型、H 模型都针对其他模型的缺点提出了一些修正意见，但其本身也可能存在一些不足的地方，所以，在实际测试工作中应该尽可能地从不同的模型中抽象出符合实际现状的、对项目有实用价值的方面，不能强行地为使用模型而使用模型。

1. 软件过程的定义

测试过程是软件过程的组成部分，明确自己的软件过程，才能明确自己的测试过程。软件生存周期是指软件从出现一个构思之日起，直到最后决定停止使用之时止，包括可行性与计划研究、需求分析、设计、实现、测试、运行与维护等阶段。

软件过程是指开发和维护软件及相关产品（如项目计划、文档、代码、手册等）的一套行为、方法、实践及变换过程。软件过程是软件生存周期的框架。

2. 测试过程与开发过程的关系

测试过程与开发过程都是软件过程的有机组成部分；测试过程与开发过程同步进行，且与开发过程相互依赖，又相互独立；开发过程、测试过程、项目管理过程以及其他支撑过程相互交织共同组成了软件过程。

3. 测试过程的活动

测试过程的活动包括计划、设计、准备、执行、评估、缺陷跟踪。

4. 测试过程的理念

（1）尽早测试

尽早测试包含两方面的含义：第一，测试人员早期参与软件项目，及时开展测试的准备工作，包括编写测试计划、制定测试方案以及准备测试用例；第二，尽早地开展测试执行工作。

- 测试可以在需求分析阶段就及早开始，在进行需求分析、产品功能设计的同时，测试人员就可以阅读、审查需求分析的结果，创建测试的准则。
- 当系统设计人员在进行系统设计时，测试人员可以了解系统是如何实现的，基于什么样的平台，这样可以设计系统的测试方案和测试计划，并事先准备系统的测试环境。
- 当设计人员在进行详细设计时，测试人员可以参与设计，对设计进行评审，找出设计的缺陷，同时设计功能、新特性等各方面的测试用例，完善测试计划。

- 在编程的同时，进行单元测试，是一种很有效的办法，可以尽快找出程序中的错误，充分的单元测试可以大幅度提高程序质量、减少成本。

由于及早地开展了测试准备工作，测试人员能够于早期了解测试的难度、预测测试的风险，从而有效提高测试效率，规避测试风险。由于及早地开展测试执行工作，测试人员可以尽早地发现软件缺陷，大大降低了 Bug 修复成本。



尽早测试并非盲目地提前展开测试活动，测试活动开展的前提是达到必需的测试就绪点。

（2）全面测试

全面测试包含两层含义：第一，对软件的所有产品进行全面的测试，包括需求、设计文档、设计代码、设计用户文档等。第二，软件开发及测试人员（有时包括用户）全面地参与到测试工作中，例如对需求的验证和确认活动，就需要开发、测试及用户的全面参与，毕竟测试活动并不仅仅是保证软件运行正确，同时还要保证软件满足了用户的需求。

在具体的测试项目的跟踪与监控过程中，可以采用周报、日报、例会，以及评审会等方式来了解测试项目的进展情况，建立、收集和分析项目的实际状态数据，对项目进行跟踪与监控，从而达到项目管理的目的。

（3）全过程测试

全过程测试包含两层含义：第一，测试人员要充分关注开发过程，对开发过程的各种变化及时做出响应，例如开发进度的调整可能会引起测试进度及测试策略的调整，需求的变更会影响到测试的执行等。第二，测试人员要对测试的全过程进行全程的跟踪，例如建立完善的度量与分析机制，通过对自身过程的度量，及时了解过程信息、调整测试策略。

（4）独立的、迭代的测试

在遵循尽早测试、全面测试、全过程测试理念的同时，应当将测试过程从开发过程中适当地抽象出来，作为一个独立的过程进行管理。时刻把握独立的、迭代测试的理念，减小因开发模型的繁杂给测试管理工作带来的不便。对于软件过程中不同阶段的产品和不同的测试类型，只要测试准备工作就绪，就可以及时开展测试工作，把握产品质量。

5. 测试过程的阶段

测试过程可分为测试项目启动阶段、测试计划阶段、测试设计阶段、测试执行阶段、测试结果的审查和分析阶段。

制定软件测试过程要注意执行的效果，将测试工作落实到实处，而不是为了走过场证明测试工作已经达到了某种程度，否则再好、再适合的过程也不能起到它的作用。

测试过程不光要制定软件测试部门内部的工作过程，更要制定与开发部门、需求部门等外部部门的接口工作的过程，一旦项目在进度、功能上有了变化要及时和测试部门沟通，不要让测试部门成为最后一个知情人。

20.4 组织和人员管理

软件测试的组织是将软件测试人员有效地组织起来、分工合作，形成一支精干的队伍，使他们发挥出最大的工作效率。组织职能是指为了实现组织的共同目标，而确定组织内各个要素及其相互关系的一系列活动的总称。简单来讲，组织职能就是设计一个组织结构并使之运转。

20.4.1 软件测试的组织

1. 测试组织的任务

测试组织的任务如下：

- 为测试项目选择合适的组织结构模式。
- 确定项目组内部的组织形式。
- 合理配备人员，明确分工和责任。
- 对项目成员的思想、心理和行为进行有效的管理，充分发挥他们的主观能动性，密切配合实现项目的目标。

2. 测试组织的管理原则

测试组织管理的原则如下。

- 尽快落实责任：从软件的生存周期来看，测试往往是指对程序的测试，但是，由于测试的依据是规格说明书、设计文档和使用说明书，如果设计有错误，测试的质量就难以保证。实际上，测试的准备工作在分析和设计阶段就开始了，在软件项目的开始就要尽早指定专人负责，让其有权去落实与测试有关的各项事宜。
- 减少接口：要尽可能地减少项目组内人与人之间的层次关系，缩短通信的路径，方便人员之间的沟通，提高工作效率。
- 责任明确、均衡：项目组成员都必须明确自己在项目组中的地位、角色和职责，各成员所负的责任不应比委任的权力大，反之亦然。

3. 测试组织的人员组成

对测试组织的人员组成的说明如下。

- 测试经理：负责测试流程、沟通、测试工具的引入、人员管理、测试计划/设计/开发及执行。
- 测试组长：负责沟通、测试工具引入、人员管理、费用/过程状态报告、测试计划/设计/开发及执行。
- 测试工程师：负责执行测试计划、进行设计/开发及执行。

4. 测试组织的规模

测试组织规模的确定方法有开发比例法、百分比法、测试程序法、任务计划法。

（1）开发比例法

开发比例法是根据开发人员的数量并按照一定的比例来确定测试工程师的数量。开发人员是指进行设计、开发、编译以及进行单元测试的人员。

对开发比例法的说明如表 20-1 所示。

表 20-1　开发比例法

开发类型	开发人员	比例	测试组规模
商业产品（大型市场）	30 人	3:2	20
商业产品（小型市场）	30 人	3:1	10
单个客户端的应用开发	30 人	6:1	5
单个客户端开发并与系统集成	30 人	4:1	7
政府部门应用开发（内部）	30 人	5:1	6
公司应用开发（内部）	30 人	4:1	7

（2）百分比法

百分比法是根据测试人员应该占到项目组人员的百分比数量进行衡量，如表 20-2 所示。

表 20-2　百分比法

开发类型	项目人员数量	测试组规模比例	测试组规模
商业产品（大型市场）	50 人	27%	13
商业产品（小型市场）	50 人	16%	8
单个客户端的应用开发	50 人	10%	5
单个客户端开发并与系统集成	50 人	14%	7
政府部门应用开发（内部）	50 人	11%	5
公司应用开发（内部）	50 人	14%	7

（3）测试程序法

测试程序法是根据测试程序数量，以及每个程序可能的执行时间，计算出人小时，再根据完成周期计算测试组规模，如表 20-3 所示。

表 20-3　测试程序法

	测试过程数目	计算因子	人小时	完成周期	测试组规模
历史记录					
新项目评估					

（4）任务计划法

任务计划法是根据历史记录中类似项目工作量，比较新项目同历史项目的工作量，历史项目再乘以相应的因子。步骤是：先将任务分解，根据历史记录乘以一个因子计算出新项目的所有任务工作量，再根据该工作量和完成周期计算测试组规模。

20.4.2 软件测试组织的职能

对软件测试组织的职能的说明如下：

- 按照组织目标和实施计划，建立合理的组织机构，包括各个管理层次和职能部门的建立。
- 按照业务性质进行分工，确定各个部门的职责范围。
- 按照所负责任给予各个部门、各管理人员相应的权利。
- 明确各部门之间、上下级之间的领导和协作关系，建立通畅的信息沟通渠道。
- 配备和使用适合工作要求的人员。

20.4.3 软件测试的组织结构

软件测试的组织结构可分为独立测试组和集中测试组。

1. 独立测试组

独立测试组的测试人员由临时人员组成，通常有 2～5 人，直接由项目经理负责管理。大型项目可以划分为几个小组，缺点是：企业没有正规的方法将测试程序、方法、相关的知识经验传递下去，测试质量难以保证。优点是：成本低，不需要对测试人员提供培训、生活保障等服务。

2. 集中测试组

企业成立专职、独立的测试部门，通常由 10～30 人组成。集中测试组为每个项目配备几个全职的测试人员。优点是：可以将相关的知识、经验传递下去。还可以由软件开发组织之外的人员或其中的独立人员组成，如转包商等。

20.4.4 软件测试组织结构的准则

软件测试组织结构的准则如下：

- 提供软件测试的快速决策能力。
- 利于合作，尤其是产品开发与测试开发之间的合作。
- 能够独立、规范、不带偏见地运作并具有精干的人员配置。
- 有利于满足软件测试与质量管理的关系。
- 有利于满足软件测试过程管理要求。
- 有利于为测试技术提供专有技术。
- 充分利用现有测试资源，特别是人。
- 对测试者的职业道德和事业产生积极的影响。

20.4.5 软件测试人员的能力要求

普通测试人员的能力应包括以下几项：

- 技术知识：包括表达、交流、协调、管理、质量意识、过程方法、软件工程等。
- 测试技能及方法：包括测试基本概念及方法、测试工具及环境、专业测试标准、工作成绩评估、熟悉所测试的产品用到的技术，并掌握测试工具、方法等相关技术等。

- 测试规划能力：包括将业务任务和技术任务相互独立、能够适应不同的测试项目、风险分析及防范、软件放行/接收准则制定、测试目标及计划、测试计划和设计的评审方法等。
- 测试执行能力：包括有成熟的测试过程管理规范、测试数据/脚本/用例、测试比较及分析、缺陷记录及处理、自动化工具。
- 测试分析、报告和改进能力：包括测试度量、统计技术、测试报告、过程监测及持续改进。

测试组织管理者的工作能力在很大程度上决定了测试工作的成功与否，测试管理是很困难的，测试组织的管理者必须具备如下能力：

- 具有了解与评价软件测试政策、标准、过程、工具、培训和度量的能力。
- 具有领导一个测试组织的能力，该组织必须坚强有力、独立自主、办事规范且没有偏见。
- 具有吸引并留住杰出测试专业人才的能力。
- 具有领导、沟通、支持和控制的能力。
- 具有提出解决问题方案的能力。
- 具有控制测试时间、质量和成本的能力。

20.5 软件配置管理

20.5.1 软件配置管理概述

1. 软件配置管理的定义

软件配置管理（Software Configuration Management，SCM）是一种标识、组织和控制修改的技术，目的是使错误降为最小并最有效地提高生产效率。软件配置管理应用于整个软件工程过程。在软件建立时变更是不可避免的，而变更加剧了项目中软件开发者之间的混乱。SCM 活动的目标就是为了标识变更、控制变更、确保变更正确实现并向其他有关人员报告变更。

软件配置管理作为 CMM2 级的一个关键域（Key Practice Area，KPA），在整个软件的开发活动中占有很重要的位置，它被用来标识变化、控制变化、保证变化被适当地发现、向其他可能有兴趣的人员报告变化。

软件配置管理的定义包括：

- 标识：识别产品的结构、产品的构件及其类型，为其分配惟一的标识符，并以某种形式提供对它们的存取。
- 控制：通过建立产品基线，控制软件产品的发布和在整个软件生命周期中对软件产品的修改，例如，它将解决那些修改会在产品的最新版本中实现的问题。
- 状态统计：记录并报告构件和修改请求的状态，并收集关于产品构件的重要统计信息，例如，它将解决修改这个错误会影响多少个文件的问题。
- 审计和审查：确认产品的完整性并维护构件间的一致性，即确保产品是一个严格定义的构件集合，例如，它将解决目前发布的产品所用的文件的版本是否正确的问题。
- 生产：对产品的生产进行优化管理。它将解决最新发布的产品应由哪些版本的文件和工具来生成的问题。
- 过程管理：确保软件组织的规程、方针和软件周期得以正确贯彻执行。它将解决要交付给

用户的产品是否经过测试和质量检查的问题。

- 小组协作：控制开发统一产品的多个开发人员之间的协作，例如，它将解决是否所有本地程序员所做的修改都已被加入到新版本的产品中的问题。
- 结构：表示产品的架构。
- 创建：支持产品的构建及其产品的附件。
- 审核：对产品及其过程的审核予以保留。
- 统计：采集与产品、过程相关的数据。
- 控制：控制产品变更的方式及时间。
- 过程：支持产品演变的管理。
- 团队协作：促进项目组开发及产品维护。

2. 软件配置管理的基本目标

软件配置管理是在贯穿整个软件生命周期中建立和维护项目产品的完整性。它的基本目标如下：

- 软件配置管理的各项工作是有计划进行的。
- 被选择的项目产品得到识别、控制并且可以被相关人员获取。
- 已识别出的项目产品的更改得到控制。
- 使相关组别和个人及时了解软件基准的状态和内容。

3. 软件配置管理的角色职责

在软件配置管理的过程中主要涉及以下的角色和分工。

（1）项目经理（Project Manager，PM）

项目经理是整个软件研发活动的负责人，他根据软件配置控制委员会的建议批准配置管理的各项活动并控制它们的进程。其具体职责为以下几项：

- 制定和修改项目的组织结构和配置管理策略。
- 批准、发布配置管理计划。
- 决定项目起始基线和开发阶段。
- 接受并审阅配置控制委员会的报告。

（2）配置控制委员会（Configuration Control Board，CCB）

配置控制委员会负责指导和控制配置管理的各项具体活动的进行，为项目经理的决策提供建议。其具体职责为以下几项：

- 定制开发子系统。
- 定制访问控制。
- 制定常用策略。
- 建立、更改基线的设置，审核变更申请。
- 根据配置管理员的报告决定相应的对策。

（3）配置管理员（Configuration Management Officer CMO）

配置管理员根据配置管理计划执行各项管理任务，定期向 CCB 提交报告，并列席 CCB 的例

会。其具体职责为以下几项：

- 软件配置管理工具的日常管理与维护。
- 提交配置管理计划。
- 负责各配置项的管理与维护。
- 执行版本控制和变更控制方案。
- 完成配置审计并提交报告。
- 对开发人员进行相关的培训。
- 识别软件开发过程中存在的问题并拟就解决方案。

（4）系统集成员（System Integration Officer，SIO）

系统集成员负责生成和管理项目的内部和外部发布版本，其具体职责为以下几项：

- 集成修改。
- 构建系统。
- 完成对版本的日常维护。
- 建立外部发布版本。

（5）开发人员（Developer）

开发人员的职责就是根据组织内确定的软件配置管理计划和相关规定，按照软件配置管理工具的使用模型来完成开发任务。

4. 软件配置管理的过程

一个软件研发项目一般可以划分为三个阶段：计划阶段、开发阶段和维护阶段。而开发阶段和维护阶段从配置管理的角度来看所涉及的活动是一致的，所以统一称为“项目开发和维护”阶段。

（1）项目计划阶段

一个项目设立之初，首先需要制订整个项目的计划，它是项目研发工作的基础。在有了总体研发计划之后，软件配置管理的活动就可以展开了，因为如果不在项目开始之初制定软件配置管理计划，那么软件配置管理的许多关键活动就无法及时有效地进行，而它的直接后果就是造成了项目开发状况的混乱并注定软件配置管理活动成为一种“救火”的行为，所以及时制订一份软件配置管理计划在一定程度上是项目成功的重要保证。

在软件配置管理计划的制订过程中，它具有以下主要流程：

- CCB 根据项目的开发计划确定各个里程碑和开发策略。
- CMO 根据 CCB 的规划，制订详细的配置管理计划，交 CCB 审核。
- CCB 通过配置管理计划后交项目经理批准，发布实施。

（2）项目开发和维护阶段

这一阶段是项目研发的主要阶段。在这一阶段中软件配置管理活动主要分为三个层面：

- 由 CMO 完成管理和维护工作。
- 由 SIO 和 DEV 具体执行软件配置管理策略。
- 变更流程。

这三个层面是彼此之间既独立又互相联系的有机整体。在这个软件配置管理过程中，它的核心流程应该是这样的：

- CCB 设定研发活动的初始基线。
- CMO 根据软件配置管理规划设立配置库和工作空间，为执行软件配置管理做好准备。
- 开发人员按照统一的软件配置管理策略，根据获得的授权资源进行项目的研发工作。
- SIO 按照项目的进度集成组内开发人员的工作成果，并构建系统，推进版本的演进。
- CCB 根据项目的进展情况，审核各种变更请求，并适时地划定新的基线，保证开发和维护工作有序的进行。

这个流程就是如此循环往复，直到项目的结束。当然，在上述的核心过程之外，还涉及其他一些相关的活动和操作流程，下面按不同的角色分工予以列出：

- 各开发人员按照项目经理发布的开发策略或模型进行工作。
- SIO 负责将各分项目的工作成果归并至集成分支，供测试或发布。
- SIO 可向 CCB 提出设立基线的要求，经批准后由 CMO 执行。
- CMO 定期向项目经理和 CCB 提交审计报告，并在 CCB 例会中报告项目在软件过程中可能存在的问题和改进方案。
- 在基线生效后，一切对基线和基线之前的开发成果的变更必须经 CCB 的批准。
- CCB 定期举行例会，根据成员所掌握的情况、CMO 的报告和开发人员的请求，对配置管理计划作出修改。

综上所述，配置管理的工作流程如图 20-1 所示。

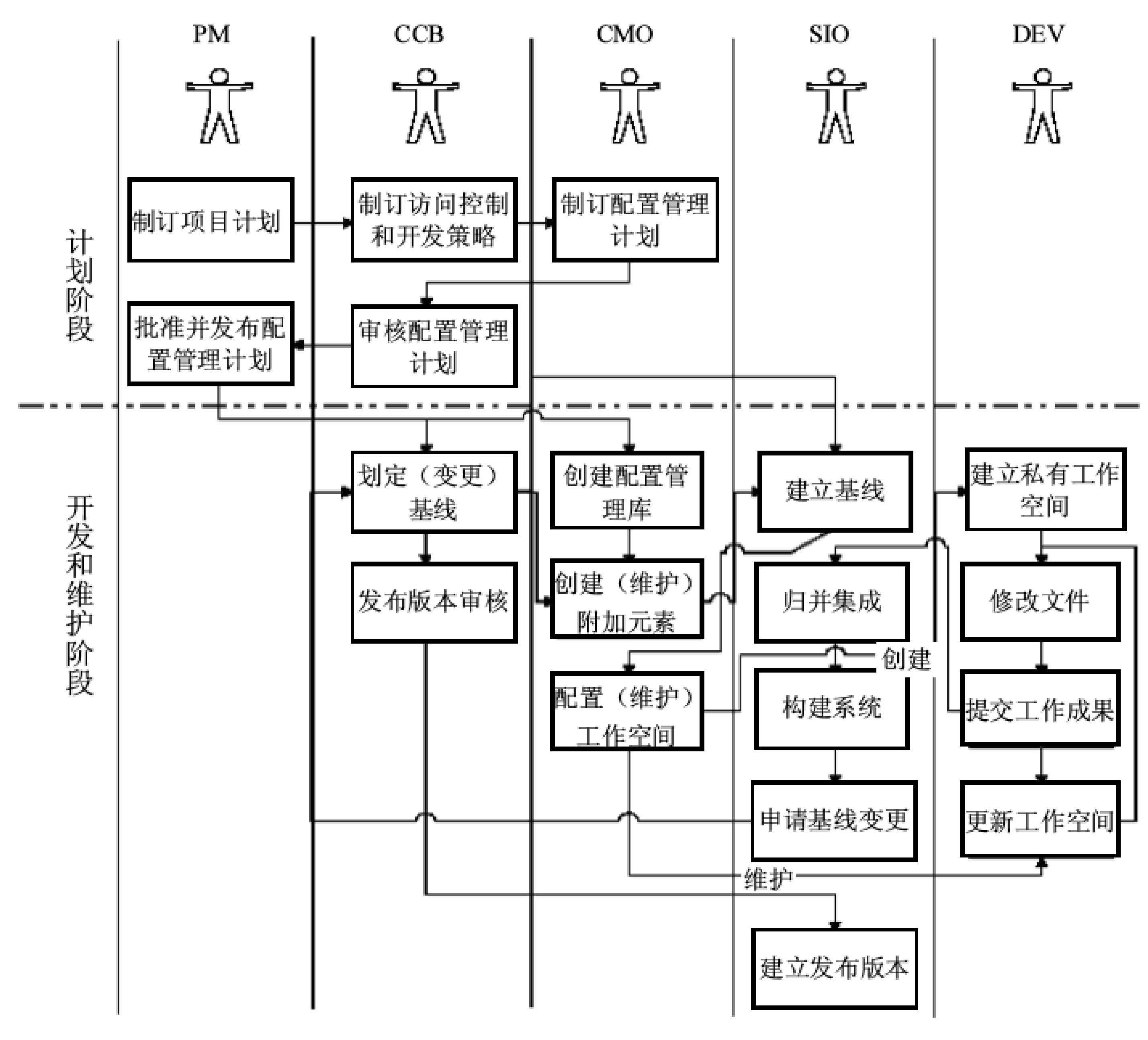


图 20-1　配置管理的工作流程图

5. 软件配置管理的关键活动

配置管理的目的是建立和维护在软件生命周期中软件产品的完整性和一致性。一般来说，软件测试配置管理包括 6 个最基本关键活动：

- 配置标识。
- 版本控制。
- 变更控制。
- 配置状态报告。
- 配置审计。
- 工作空间管理。

（1）配置标识

配置标识是配置管理的基础，也是制订配置管理计划的重要内容。所有配置项的操作权限都应当严格管理，其基本原则是：所有基线配置项向测试人员开放读取权限；而非基线配置项向测试组长、项目经理及相关人员开放。

配置标识主要是标识测试样品、测试标准、测试工具、测试文档（包括测试用例）、测试报告等配置项的名称和类型。所有配置项都应按照相关规定统一编号，按照相应的模板生成，并在文档中的规定章节（部分）记录对象的标识信息。在引入软件配置管理工具进行管理后，这些配置项都应以一定的目录结构保存在配置库中，这样使得测试相关人员能方便地知道每个配置项的内容和状态。

（2）版本控制

版本控制是软件配置管理的核心功能。版本控制的目的是按照一定的规则保存配置项的所有版本，避免发生版本丢失或混淆等现象，并且可以快速准确地查找到配置项的任何版本。所有置于配置库中的元素都应自动予以版本的标识，并保证版本命名的惟一性。版本在生成过程中，自动依照设定的使用模型自动分支、演进。除了系统自动记录的版本信息以外，为了配合软件开发流程的各个阶段，还需要定义、收集一些元数据来记录版本的辅助信息和规范开发流程，并为今后对软件过程的度量做好准备。当然如果选用的工具支持的话，这些辅助数据将能直接统计出过程数据，从而方便软件过程改进活动的进行。

对于配置库中的各个基线控制项，应该根据其基线的位置和状态来设置相应的访问权限。一般来说，对于基线版本之前的各个版本都应处于被锁定的状态，如果需要对它们进行变更，则应按照变更控制的流程来进行操作。

（3）变更控制

变更控制的目的并不是控制和限制变更的发生，而是对变更进行有效的管理，确保变更有序的进行。变更管理的一般流程如下：

- （获得）提出变更请求。
- 由 CCB 审核并决定是否批准。
- （被接受）修改请求分配人员，提取 SCI，进行修改。
- 复审变化;
- 提交修改后的 SCI。

- 建立测试基线并测试。
- 重建软件的适当版本。
- 复审（审计）所有 SCI 的变化。
- 发布新版本。

（4）配置状态报告

配置状态报告是根据配置项操作数据库中的记录，向管理者报告软件测试工作的进展情况。这样的报告应该是定期进行的，并尽量通过 CASE 工具自动生成，利用数据库中的客观数据来真实地反映各配置项的情况。

配置状态报告应根据报告着重反映当前基线配置项的状态，以作为对开发进度报告的参照。同时也能从中根据开发人员对配置项的操作记录来对开发团队的工作关系进行一定的分析。配置状态报告应该包括以下主要内容：

- 定义配置状态报告的形式、内容和提交方式。
- 确认过程记录和跟踪问题报告，更改请求，更改次序等。
- 确定测试报告提交的时间与方式。
- 配置库结构和相关说明。
- 开发起始基线的构成。
- 当前基线位置及状态。
- 各基线配置项集成分支的情况。
- 各私有开发分支类型的分布情况。
- 关键元素的版本演进记录。
- 其他应予报告的事项。

（5）配置审计

配置审计的主要作用是作为变更控制的补充手段，来确保某一变更需求已被切实地执行和实现。配置审计包括以下主要内容：

- 确定审计执行人员和执行时机。
- 确定审计的内容与方式。
- 确定发现问题的处理方法。
- 制订项目的配置计划。
- 对配置项进行标识。
- 对配置项进行版本控制。
- 对配置项进行变更控制。
- 定期进行配置审计。
- 向相关人员报告配置的状态。

（6）工作空间管理

在引入了软件配置管理工具之后，所有开发人员都会被要求把工作成果存放到由软件配置管理工具所管理的配置库中去，或是直接工作在软件配置管理工具提供的环境之下，所以为了让每个开发人员和各个开发团队能更好地分工合作，同时又互不干扰，对工作空间的管理和维护也成为了

软件配置管理的一个重要活动。

20.5.2 软件配置管理要求

软件配置管理的要求主要有如下 10 项内容：

- 配置管理适用的范围，包括 SNTC 部门的全部工作产品、研发中心各个部门的评审记录。
- 配置管理下的项至少应包括工作计划、工作任务、工作周报、各种会议记录、经评审确认的工作产品、评审记录等。
- 配置管理命名规则。
- 配置库文件目录结构，其中配置库文件目录结构如图 20-4 所示。
- 角色和责任。
- 目录添加/修改/删除流程。
- 配置项的添加/修改/删除流程。
- 配置项的发布。
- 配置管理文档的保存。
- 配置库备份。

一级	二级	三级	四级	说明
SNTC				有一个文件：配置项列表.xls
	工作计划			没有文件
				每月内的工作计划
	任务和周报			没有文件
				每周的工作任务和工作周报
	会议纪要			没有文件
				每月内的会议记要
	内部成果			没有文件
		成果简报		此成果的全部文件（可以有子目录）
	监督记录			没有文件
				每月的监督记录/执行情况报告
	其他文件			没有文件
		内容简报		此内容的全部文件（可以有子目录）
STB				没有文件
	评审记录			没有文件
		内容简报		此评审的全部过程文档
公司规程				此目录可以存储单文件规程
	[规程简称]			该规程的全部文档
公共文件				此目录可以存储单文件文档
	[公司简称]			该内容的全部文档

图 20-4　配置库文件目录结构表

对于元素的要求如下：

- 要记录元素的版本及其差异、差异的原因。
- 确定构成配置及配置版本的组件群。
- 标识出产品的基线及其外延产品，确定表示项目组件群及附件项目的环境。

对于机构的要求如下：

- 要通过表示产品组件库的系统模型来模拟产品的结构。
- 标明组件、版本、配置的界面使之可以重用。
- 确定及维护组件间的关系，选择兼容的组件使之形成有效的、一致的产品版本。

对构建的要求如下：

- 要容易创建产品的手段。
- 能随时静态分析产品的现状。
- 通过减少组件的堆积和节省区间来优化系统创建的机制。
- 进行更改分析以预测因更改而导致的细小分化的手段。
- 随时都能对产品的任何部分、在任何阶段容易得到更新。

对于审核的要求如下：

- 要所有更改的历史记录。
- 所有与产品相关的组件与其演变的追溯性。
- 完成任务的所有细节的日志。

对于统计的要求是：要统计记录的机制、产品现状的检验，有关产品和过程的所有方面的报告能较易产生。

对于控制的要求如下：

- 要为避免不必要的变更或变更冲突对系统中的组件的获取予以控制，对于更改要求的表格及问题报告形成在线支持。
- 错误查找的手段及何时对何人会产生什么影响。
- 在不同但相关的产品版本之间以受控的方式进行更改告知。
- 将产品进行分割的手段以限制更改影响。

对于过程的要求如下：

- 要对生命周期模型及组织方针予以支持。
- 确定要完成的任务及如何完成、何时完成的能力。
- 要有将相干的事务的信息在适当的人员之间进行沟通的能力。
- 要有将产品的经验文档化的手段。

对于团队协作的要求如下：

- 个人和小组的工作区间。
- 在汇合时产生冲突的解决办法。
- 对产品的创建及其维护予以支持的手段。

20.6 软件缺陷管理

软件缺陷管理是在软件生命周期中为确保缺陷被跟踪和管理所进行的活动。

1. 缺陷管理的目标

缺陷管理的目标是：确保每个被发现的缺陷都能够被解决；收集缺陷数据并根据分析和统计缺陷、排除缺陷以及预防缺陷等步骤达到有效地减少软件产品的缺陷数。

2. 缺陷分类

为了对缺陷进行管理，首先是了解缺陷、对缺陷进行分类，通过对缺陷进行分类，可以迅速找出哪一类缺陷的问题最大，然后集中精力预防和排除这一类缺陷。缺陷类型如表 20-5 所示。

表 20-5　缺陷分类

编号	缺陷类型	描述
01	F-功能	如逻辑、指针、循环、递归、功能等缺陷
02	G-语法	拼写、标点符号、打字
03	A-赋值	如声明、重复命名、作用域
04	I-接口	与其他组件、模块或设备驱动程序、调用参数、控制块或参数列表相互影响的缺陷
05	B-联编打包	由于配置库、变更管理或版本控制引起的错误
06	D-文档	需求、设计类文档
07	U-用户接口	人机交互特性：屏幕格式、确认用户输入、功能有效性
08	P-性能	不满足系统可测量的属性值，如执行时间、事务处理速率等
09	N-标准	不符合各种标准的要求，如编码标准、设计符号等
10	E-环境	设计、编译、其他支持系统的问题
11	W-误解	开发环境的错误或者纯粹由于缺乏解决问题的相关技术，这类 Bug 共同的特点是都来自于开发人员

3. 缺陷管理

（1）收集缺陷

缺陷管理的第一步是收集缺陷的数据，如图 20-6 所示。

<table>
<tr><td>日期</td><td>编号</td><td>状态</td><td>类型</td><td>缺陷来源</td><td>排队阶段</td><td>修改时间</td><td>修复缺陷</td></tr>
<tr><td rowspan="2"></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="7">描述：</td></tr>
<tr><td rowspan="2"></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="7">描述：</td></tr>
<tr><td rowspan="2"></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="7">描述：</td></tr>
</table>

图 20-6　收集缺陷

（2）分析缺陷来源

分析缺陷来源的示意图如表 20-7 所示。

表 20-7　分析缺陷来源

缺陷来源	描述
需求	由于需求的问题引起的缺陷
构架	由于构架的问题引起的缺陷
设计	由于设计的问题引起的缺陷
编码	由于编码的问题引起的缺陷
测试	由于测试的问题引起的缺陷
集成	由于集成的问题引起的缺陷

（3）排除修复缺陷

排除修复缺陷的示例如表 20-8 所示。

表 20-8　排除修复缺陷

排除缺陷	描述	缺陷严重等级	发现人	日期	修复人	日期
需求	在需求阶段发现的缺陷					
构架	在构架阶段发现的缺陷					
设计	在设计阶段发现的缺陷					
编码	在编码阶段发现的缺陷					
测试	在测试阶段发现的缺陷					

20.7　变更请求管理

1. 变更请求的定义

变更伴随着软件开发的各个阶段。软件开发过程中的变更可以从两个侧面来描述，一个是对软件开发过程之中工件（如需求设计文档、设计模型、代码及测试脚本等）的变更；另一方面是驱动工件变更的理由（如缺陷修正、新功能添加等）。这种驱动软件工件变更的理由就是变更请求。

变更请求是项目管理的重要数据之一，通过对这些数据的统计分析可以进行量化的项目管理。

2. 变更请求的管理

变更请求管理是使软件开发成本降低的最大因素之一，随着对软件开发的要求越来越高，变更量也越来越多，开发人员必须迅速解决变更问题。变更请求管理就需要建立合理的变更流程。实施变更请求管理流程的基本步骤如下：

（1）确定变更请求管理流程执行的范围，然后制定响应的变更流程。

（2）制定变更管理流程的模型。

（3）决定团队各个角色在流程实施中所起的作用。

（4）确定实施计划及开始实施日程。

（5）部署变更请求管理系统。

（6）不断强化变更管理流程。

为了保证整个项目开发的成功，变更请求管理应明确如下问题：

- 哪些需求发生了变化？应可提供具有各种重要特征的变更请求信息，且对各种变更请求在处理完毕之前的内容能及时调整，并保证各种请求的信息绝对不能丢失。
- 这些需求变化后，对测试工作会产生哪些影响？包括会不会影响测试用例？会不会影响到测试方案？会不会影响到测试计划？应通过对缺陷及其他各种变更的登记、保管、跟踪、解析，达到团队之间的各种变化信息的共有、安全而可靠地高质量变更信息管理系统。
- 需求变化后对工作进度产生多大的影响？有效地跟踪各种变更，对管理人员提出各种变更状况的查询请求，做到快而准地提供信息。
- 不同变更请求之间的连接关系。对项目整体发展状况，提供宏观及定量的分析，从而能合理分配项目开发人员的工作、合理制订项目的计划、合理管理项目各种请求实施的优先级。
- 统计各种项目指标数据，项目管理人员就可以进行更加科学、量化的管理、规划、调配、监控，保证项目如期的进行。

20.8 进度管理

项目的进度管理是一个动态的过程，需要不断调度、协调，保证项目的均衡发展，实现项目整体的动态平衡。

项目的进度管理主要是通过阶段、关键路径的控制并借助工具来实现，同时要把握好进度与质量、成本的关系，以及充分了解进度的数量和质量的双重特性。

1. 影响测试项目进度的因素

（1）人员、预算变更对进度的影响

有时某方面的人员不够到位，或者在多个项目的情况下某方面的人员中途被抽到其他项目、身兼多个项目、在别的项目不能自拔无法投入本项目，从而对进度造成影响。预算的变更会影响某些资源的变更，从而对进度造成影响。

（2）低估环境因素对进度的影响

企业高级项目主管和项目经理也经常低估用户环境、行业环境、组织环境、社会环境、经济环境，既有主观的原因，也会有客观的原因。对项目环境的了解程度不够，造成没有做好充分的准备，从而对进度造成影响。

（3）项目状态信息收集对进度的影响

由于项目经理的经验或素质原因，对项目状态信息收集的掌握不足，及时性、准确性、完整性比较差，从而对进度造成影响。

（4）执行计划的严格程度对进度的影响

没有把计划作为项目过程行动的基础，而是把计划放在一边，比较随意去做，从而对进度造成影响。

（5）计划变更调整的及时性对进度的影响

计划的制订需要随着项目的进展进行不断细化、调整、修正、完善。计划变更调整不及时从而对进度造成影响。

2. 测试项目的进度控制措施

（1）项目进度控制的前提

项目进度控制的前提是有效的项目计划和充分掌握第一手实际信息，在此前提下，通过实际值与计划值进行比较，检查、分析、评价项目进度。通过沟通、肯定、批评、奖励、惩罚、经济等不同手段，对项目进度进行监督、督促、影响、制约。及时发现偏差，及时予以纠正；提前预测偏差，提前予以预防。必须落实项目团队之内或之外进度控制人员的组成，明确具体的控制任务和管理职责。

（2）项目进度控制的主要手段

从进度控制内容上看，进度控制主要表现在组织管理、技术管理和信息管理等几个方面。

20.9 风险管理

风险管理是指为了更好地达到项目的目标，识别、分配、应对项目生命周期内风险的科学与艺术。

20.9.1 软件风险的基本概念

软件风险是指在软件开发过程中遇到的预算、进度、开发不成功等方面的问题引起损失的可能性。

软件测试的风险是指软件测试过程出现的或潜在的问题，造成的原因主要是测试计划的不充分、测试方法有误或测试过程的偏离，造成测试的补充以及结果不准确。测试的不成功导致软件交付潜藏着问题，一旦在运行时爆发，会导致软件失败。

软件测试风险主要是对测试计划执行的风险分析与制定要采取的应急措施，降低软件测试产生的风险造成的危害。

测试计划的风险一般是指测试进度滞后或出现非计划事件，当测试计划风险发生时，可能采用的应急措施有：缩小范围、增加资源、减少过程等措施。

- 缩小范围：决定在后续的发布中，实现较低优先级的特性。
- 增加资源：请求用户团队为测试工作提供更多的用户支持。
- 减少过程：在风险分析过程中，确定某些风险级别低的特征测试或少测试。

软件项目的风险一般体现在以下 5 个方面：需求、计划编制风险、组织和管理风险、开发环境风险、设计和实现风险。

1. 需求风险

需求风险包括如下内容：

- 范围风险：与范围变更有关的风险。
- 外部可预测风险：市场风险（原材料可利用性、需求）、日常运作（维修需求）、环境影响、社会影响、货币变动、通货膨胀、税收。
- 外部不可预测风险：规章（不可预测的政府干预）、自然灾害。
- 内部非技术风险：战略风险（公司的经营战略发生了变化）、管理风险（公司管理人员是否成熟等）。
- 需求定义欠佳，而进一步的定义会扩展项目范畴。
- 添加额外的需求。
- 产品定义含混的部分比预期需要更多的时间。
- 在做需求中用户参与不够。
- 缺少有效的需求变化管理过程。

2. 计划编制风险

计划编制风险包括如下内容：

- 计划、资源和产品定义全凭用户或上层领导口头指令，并且不完全一致。
- 计划不能现实，只能算是期望状态。
- 计划基于使用特定的小组成员，而那个特定的小组成员其实指望不上。
- 产品规模（代码行数、功能点、与前一产品规模的百分比）比估计得要大。
- 完成目标日期提前，但没有相应地调整产品范围或可用资源。
- 涉足不熟悉的产品领域，花费在设计和实现上的时间比预期得要多。
- 没有按照要求的技术性能和质量水平完成任务。
- 没有在预算的时间范围内完成任务。
- 没有在预算的成本范围内完成任务。

3. 组织和管理风险

组织和管理风险包括如下内容：

- 仅由管理层或市场人员进行技术决策，导致计划进度缓慢，计划时间延长。
- 员工离职。
- 低效的项目组结构降低生产率。
- 管理层审查决策的周期比预期的时间长。
- 预算削减，打乱项目计划。
- 缺乏必要的规范，导致工作失误与重复工作。
- 非技术的第三方的工作（预算批准、设备采购批准、法律方面的审查、安全保证等）时间比预期得延长。
- 作为先决条件的任务（如培训及其他项目）不能按时完成。
- 开发人员和管理层之间关系不佳，导致决策缓慢，影响全局。

- 缺乏激励措施，士气低下，降低了生产能力。
- 某些人员不熟悉软件工具和环境。
- 项目后期加入新的开发人员，需要进行培训并逐渐与现有成员沟通，从而使现有成员的工作效率降低。
- 由于项目组成员之间发生冲突，导致沟通不畅、设计欠佳、接口出现错误和额外的重复工作。
- 不适应工作的成员没有调离项目组，影响了项目组其他成员的积极性。
- 没有找到项目急需的具有特定技能的人。

4. 开发环境风险

开发环境风险包括如下内容：

- 设施未及时到位。
- 开发工具未及时到位。
- 开发工具不如期望得那样有效，开发人员需要时间创建工作环境或者切换新的工具。
- 新的开发工具的学习期比预期得长，内容繁多。

5. 设计和实现风险

设计和实现风险包括如下内容：

- 设计质量低下，导致重复设计。
- 用户对于最后交付的产品不满意，要求重新设计和重做。
- 用户的意见未被采纳，造成产品最终无法满足用户要求，因而必须重做。
- 用户提供的组件质量欠佳，导致额外的测试、设计和集成工作，以及额外的用户关系管理工作。
- 用户没有或不能参与规划、原型和规格阶段的审核，导致需求不稳定和产品生产周期的变更。
- 一些必要的功能无法使用现有的代码和库实现，开发人员必须使用新的库或者自行开发新的功能。
- 代码和库质量低下，导致需要进行额外的测试、修正错误或重新制作。
- 过高估计了增强型工具对计划进度的节省量。
- 分别开发的模块无法有效集成，需要重新设计或制作。
- 没有严格要求与现有系统兼容，需要进行比预期更多的测试、设计和实现工作。
- 要求与其他系统或不受本项目组控制的系统相连，导致无法预料的设计、实现和测试工作。
- 在不熟悉或未经检验的软件和硬件环境中运行所产生的未预料到的问题。
- 开发一种全新的模块将比预期花费更长的时间。
- 依赖正在开发中的技术将延长计划进度。

20.9.2 风险识别和分析

1. 风险识别

风险识别是指确定何种风险事件可能影响项目，是在项目开始，还是在项目阶段中间。风险识别包括确定风险的来源、风险产生的条件、描述其风险特征和确定哪些风险事件有可能影响本项目。风险识别不是一次就可以完成的事情，应当在项目的各个阶段进行风险识别工作。要识别风险，首先需要了解在软件开发的各个阶段都有可能发生的风险。

（1）需求分析阶段

在需求分析阶段可能发生的风险事件如下：

- 项目目标不清。
- 项目范围不明确（范围太大、太小都不可以）。
- 用户参与少或和用户沟通少。
- 对业务了解不够。
- 对需求了解不够。
- 没有进行可行性研究。

（2）设计阶段

在设计阶段可能发生的风险事件如下：

- 项目队伍缺乏经验，缺乏有经验的系统分析员。
- 没有变更控制计划，以至于变更没有依据，偏离用户需求。
- 仓促计划带来进度方面的风险。
- 由于设计人员的疏忽某个功能没有考虑进去。

（3）实施阶段

在实施阶段可能发生的风险事件如下：

- 开发环境没有具备好。
- 设计错误带来的实施困难。
- 程序员开发能力差，或程序员对开发工具不熟。
- 项目范围改变。
- 项目进度改变。
- 在一个项目内软件开发工作有一定的连续性，若需要移交和交接，有时人员离开会对项目的影响很大。
- 开发团队内部沟通不够，导致程序员对系统设计的理解上有偏差。
- 没有有效的备份方案。
- 没有切实可行的测试计划。
- 测试人员经验不足。

（4）系统验收试运行阶段

在系统验收试运行阶段可能发生的风险事件如下：

- 测试未按计划完成，发布日期推迟。
- 交付日期的滞后，耗尽了所有的资源。
- 质量差，客户不满意。
- 资金不能回收。

2. 软件风险分析

风险分析主要涉及发生问题的可能性有多大、如果问题发生了会有什么后果。通常风险分析包括以下几项内容。

- 风险标识：表示风险事件的惟一标识。
- 风险问题：风险问题发生现象的简单描述。
- 发生可能性：风险发生可能性的级别。
- 影响的严重性：风险影响的严重性的级别。
- 风险预测值：风险发生可能性与风险影响的严重性的乘积。
- 风险优先级：风险预测值从高向低的排序。

综上所述，软件风险分析的目的是：确定测试对象、确定优先级以及测试深度。在测试计划阶段，可以用风险分析的结果来确定软件测试的优先级。对每个测试项和测试用例赋予优先代码，将测试分为高、中和低的优先级类型，这样可以在有限的资源和时间条件下，合理安排测试的覆盖度与深度。

（1）风险应对方法

PMBOK 提到三种风险应对方法，即避免、减轻和接受。

- 避免：通过分析找出发生风险事件的原因，通过消除这些原因来避免一些特定的风险事件发生。
- 减轻：通过降低风险事件发生的概率或得失量来减轻对项目的影响，也可以采用风险转移的方法来减轻风险对项目带来的影响。项目预算中考虑应急储备金是另一种降低风险影响的方法。
- 接受：接受风险造成的后果，如为了避免自然灾害造成的后果，在一个大的软件项目中考虑了异地备份中心。

（2）风险量化

风险量化涉及对风险及风险的相互作用的评估，是衡量风险概率和风险对项目目标影响程度的过程。风险量化的基本内容是确定哪些事件需要制定应对措施。

（3）风险应对计划制订

风险应对计划制订是针对风险量化的结果，为降低项目风险的负面效应制订风险应对策略和技术手段的过程。风险应对计划依据风险管理计划、风险排序、风险认知等，得出风险应对计划、

剩余风险、次要风险以及为其他过程提供的依据。

（4）风险监控

风险监控涉及整个项目管理过程中的风险进行应对。该过程的输出包括应对风险的纠正措施以及风险管理计划的更新。

20.9.3 软件项目风险管理模型

软件项目风险管理模型主要有 Boehm 模型、CRM 模型和 SERIM 模型。

1. Boehm 模型

模型：RE=P*L。

其中：RE 表示风险或者风险所造成的影响；P 表示令人不满意的结果所发生的概率；L 表示糟糕的结果会产生的破坏性的程度。

2. CRM 模型

CRM（Continuous Risk Management）模型的风险管理原则是：不断地评估可能造成恶劣后果的因素；决定最迫切需要处理的风险；实现控制风险的策略；评测并确保风险策略实施的有效性。CRM 模型要求在项目生命期的所有阶段都关注风险识别和管理，它将风险管理划分为 5 个步骤：风险识别、分析、计划、跟踪、控制。

3. SERIM 模型

SERIM（Software Engineering Risk Model）从技术和商业两个角度对软件风险管理进行剖析，考虑的问题涉及开销、进度、技术性能等。它还提供了一些指标和模型来估量和预测风险，由于这些数据来源于大量的实际经验，因此具有很强的说服力。

20.10 成本管理

1. 成本管理的主要内容

成本管理的主要内容有：资源计划、成本估算、成本预算、成本控制。

（1）资源计划

资源计划用于确定完成项目各个活动中需要资源（人、设备、材料）的种类，以及每种资源的需要量。

（2）成本估算

成本估算是为完成项目各项任务所需要的资源成本的近似估算。

（3）成本预算

成本预算是将总投资估算分配落实到各个单项工作上。项目成本预算是进行项目成本控制的基础，它是将项目的成本估算分配到项目的各项具体工作上，以确定项目各项工作和活动的成本定

额，制定项目成本的控制标准，规定项目意外成本的划分与使用规则的一项项目管理工作。

（4）成本控制

成本控制用于控制预算的变更。成本控制的每一部分都有输入、工具技术和输出：首先是根据历史信息、范围陈述、资源池描述、组织方针和活动持续期预计，利用专家判断、选择性鉴定和项目管理软件，得到资源需求文档；成本估算是根据资源需求说明、资源费用、活动持续期估计、估计发布、历史信息及账目表、风险，利用相似估计、参变模型、自底向上估计、计算机化工具、其他成本估计方法，得出成本估计、支持细节和成本管理计划。成本预算核定是根据成本估算、项目进度和风险管理计划，利用成本预算工具和技术，得到项目成本基线（成本基线是基于有限时间的预算，常用来测量监视项目成本性能）。成本控制是根据成本基线、性能报告、需求变化和风险管理计划，采用成本变化管理系统、性能测量、附加计划和计算机化工具，得到修正的成本估计、预算变动、纠正活动和完成估计。

2. 成本控制的原则

（1）投资最优化原则

信息工程项目投资控制的根本目的在于通过各种成本管理手段，在保证项目进度和质量的前提下不断降低信息工程的项目成本，从而实现目标成本最优化的要求。在实行成本最优化原则时，应注意降低成本的可能性和合理的成本最优化。一方面挖掘各种降低成本的能力，使可能性变为现实；另一方面要从实际出发，制定通过主观努力可能达到合理的最优成本水平。

（2）全面成本控制原则

全面成本管理是所有承建单位、项目参与人员和全过程的管理，亦称“三全”管理。项目成本的全员控制有一个系统的实质性内容，包括各承建单位、建设单位、监理单位等的责任，应防止成本控制人人有责、人人不管。项目成本的全过程控制要求成本控制工作要随着项目实施进展的各个阶段连续进行，既不能疏漏，又不能时紧时松，应使信息工程项目成本自始至终置于有效的控制之下。

（3）动态控制原则

信息工程项目是一次性的，成本控制应强调项目的中间控制，即动态控制，因此实施准备阶段的成本控制是根据实施组织设计的具体内容确定成本目标、编制成本计划、制订成本控制的方案，为今后的成本控制作好准备；在实施阶段，根据已经制订的成本控制方案进行动态纠偏，并根据项目的实施情况调整成本控制方案；而竣工阶段的成本控制，由于成本盈亏已基本定局，即使发生了纠差，也已来不及纠正。

在管理过程中，不能简单地把成本控制仅仅理解为将信息工程项目实际发生的成本控制在计划投资的范围内，而应当认识到，成本控制是与质量控制和进度控制同时进行的，它是针对整个信息工程项目目标系统所实施的控制活动的一个组成部分，在实现成本控制的同时需要兼顾质量和进度目标。

（4）目标管理原则

目标管理的内容包括：目标的设定和分解，目标的责任到位和执行，检查目标的执行结果，

评价目标和修正目标，形成目标管理的计划、实施、检查、处理循环，即 PDCA 循环。

在项目实施过程中，承建单位、建设单位在肩负成本监督控制责任的同时，享有成本监督控制的权力，同时承建单位的项目经理要对各小组在成本控制中的业绩进行定期的检查和考评，实行有奖有罚。只有真正做好责、权、利相结合的成本控制，才能收到预期的效果。

3. 成本管理要点

工程项目成本的构成一般可以划分为：工程前期费用、咨询/设计费用、工程费用、第三方工程测试费用、工程验收费用、系统运行维护费用、风险费用、其他费用。成本管理的要点如下：

- 项目实际成本不超过项目计划投资。
- 应十分重视项目前期（设计开始前）和设计阶段的投资控制工作。
- 以动态控制原理为指导进行投资计划值与实际值的比较。
- 可采取组织、技术、经济、合同措施。
- 有必要进行计算机辅助投资控制。

4. 技术经济分析和步骤

（1）技术经济分析的特点

根据有关资料统计，在同样能满足功能要求的前提下，技术经济合理的设计可以降低工程造价的 15%～20%，因此，搞好方案的技术经济分析，就成为建设项目设计阶段优选方案和控制造价的重点环节。技术经济分析的特点如下。

- 综合性：技术经济分析学是根据现代科学技术和国民经济发展的需要，逐渐地从自然科学和社会科学的发展过程中交叉形成发展起来的一门综合性边缘科学。
- 系统性：它研究的对象大多是由若干个相互联系的单元所组成的整体，因此要具备系统分析的思想方法和工作方法，要从整体着眼，周密地分析各个因素和环节，取得科学依据，实现总体优化。
- 实用性：它是一门实践性很强的应用科学，其主要的研究对象是技术方案，进行具体评价，为采用的方案提出技术经济效果的论据，如工程优化设计、系统方案选择等。
- 数据化：技术经济学采用了许多定量分析方法，把各种有关因素定量化，通过定量计算，进行分析比较。由于计算机和数学方法的迅速发展，定量分析范围日益扩大。除去环境保护、政治因素、学术发展等社会因素目前还只能做定性分析外，大量问题均可数据化，因此定性分析与定量分析相结合、以量化为主是技术经济分析的一大特点。

（2）技术经济分析的方法步骤

技术经济分析一般包括以下几个步骤：确定目标、调查研究、方案评价。

5. 总成本费用的估算概念

成本估算涉及计算完成项目所需各资源（人、材料、设备等）成本的近似值。

成本估算涉及的是对可能数量结果的估计，即承建单位为提供产品或服务的花费是多少。而定价是一个商业决策，即承建单位为它提供的产品或服务索取多少费用，成本估算只是定价要考虑的因素之一。在进行估算时应注意以下几点：

- 当项目在一定的约束条件下实施时，价格的估算是一项重要的因素。
- 费用估算应该与工作质量的结果相联系。
- 费用估算过程中，也应该考虑各种形式的费用交换，如在多数情况下，延长工作的延续时间通常是与减少工作的直接费用联系在一起的，相反，追加费用将缩短项目工作的延续时间。

因此，在费用估计的过程中必须考虑附加的工作对期望工期缩短的影响。

6. 成本估算主要依赖的资料

成本估算主要依赖的资料有：工作分解结构、资源需求计划、资源价格、工作的延续时间、历史信息、财务图表等。

7. 成本预算的控制

成本预算编制是一项十分细致复杂的工作，计算中难免出现一些疏漏和错误，为此必须搞好审核工作，审核的重点是：编制依据是否符合规定、造价及各项经济指标是否合理、单位工程有无漏项、说明是否全面，并做到内容完整、造价正确、经济指标及主要设备合理、软件配置合理。预算的审核本身也是对成本控制的一种方法，目的是发现并纠正错误，从而起到控制成本和造价的目的。成本控制主要关心的是影响改变费用线的各种因素、确定费用线是否改变以及管理和调整实际的改变。成本控制包括：监控费用的执行情况以确定与计划的偏差；确使所有发生的变化都被准确记录在费用线上；避免不正确的、不合适的或者无效的变更反映在费用线上；建设单位权益改变的各种信息。

8. 成本控制的管理工作

成本控制的管理工作主要有如下方面：

- 参与项目总投资目标的分析、论证、审核（在可行性研究的基础上，再作详细的分析、论证）。
- 对项目总投资切块、分解规划结果进行审核、确认，并在项目实施过程中监督其执行；在项目实施过程中，若有必要，及时提出调整总投资切块、分解规划的建议。
- 审核承建单位编制的项目实施各阶段、各年、季度等阶段性资金使用计划，并控制其执行，必要时，对上述计划提出调整建议。
- 审核工程估算、预算、标底等。
- 在项目实施过程中，按阶段（月、季）进行投资计划值与实际值的比较，并按阶段（每月、季、年）提交各种投资控制报表和报告。
- 对设计、实施、开发方法、器材和设备等多个方面作必要的技术经济比较，以能够提出有效的建议，从而挖掘出节约投资、提高项目经济效益的潜力。
- 审核招/投标文件和合同文件中有关投资的条款。
- 审核各类工程付款单。

测试费用从经济学的角度考虑就是确定需要完成多少测试，以及进行什么类型的测试。经济学所做的判断，确定了软件存在的缺陷是否可以接受？如果可以接受，能承受多少？

测试费用的有效性，可以用测试费用的质量曲线来表示，如图 20-2 所示。随着测试费用的增

加，发现的缺陷也会越多，两线相交的地方是过多测试开始的地方，这时，排除缺陷的测试费用超过了缺陷给系统造成的损失费用。测试的策略不再主要由软件人员和测试人员来确定，而是由商业的经济利益来决定。

对风险测试得过少，会造成软件的缺陷和系统的瘫痪；而对风险测试得过多，就会使本来没有缺陷的系统进行没有必要的测试，或者是对轻微缺陷的系统所花费的测试费用远远大于它给系统造成的损失。

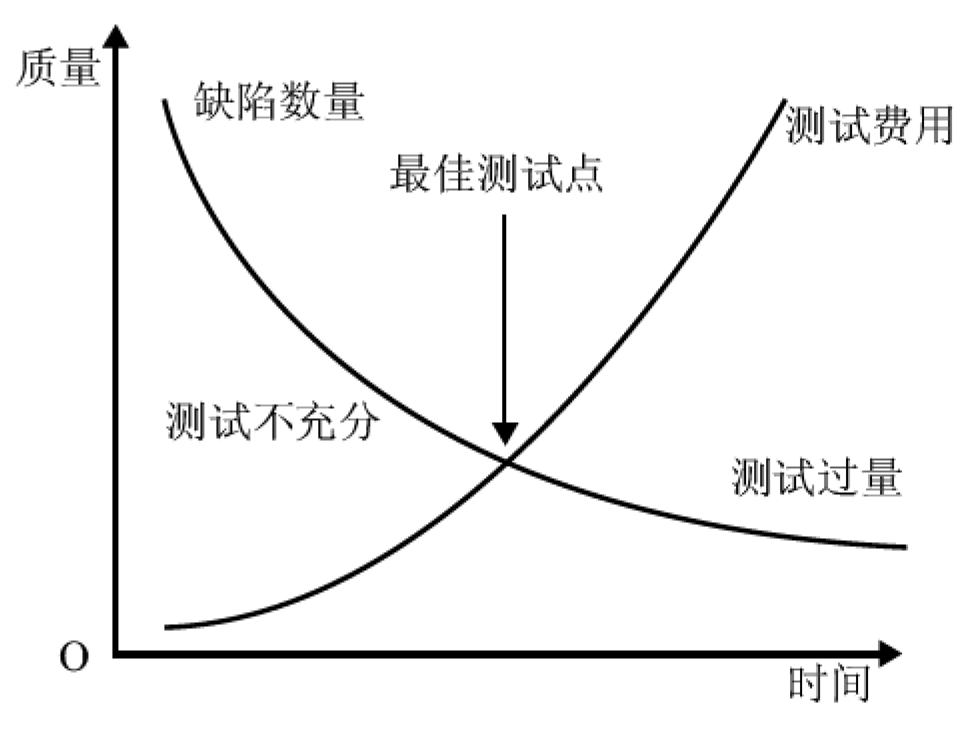


图 20-2　测试费用的质量曲线

参考文献

[1] 黎连业，李淑春．管理信息系统设计与实施．北京：清华大学出版社，1998．11

[2] 黎连业等．软件能力成熟度模型(CMM)与软件开发技术．北京：北京航空航天大学出版社，2003．05

[3] 黎连业等．信息系统工程监理工程师手册．北京：电子工业出版社，2006．11

[4] 黎连业等．计算机管理信息系统设计与实现．北京：学苑出版社，1994．12

[5] 柳纯录，黄子河，陈渌萍．软件评测师教程．北京：清华大学出版社，2005．09

[6] 佟伟光．软件测试．北京：人民邮电出版社，2008．05

[7] 中华人民共和国国家标准．评价者用的过程．GB / T18905

[8] 广东省软件公共技术中心技术资料

[9] 陕西华商数码信息股份有限公司技术资料

[10] 软件测试用例模板大全的技术资料

[11] Webmaster 中《白盒测试工具 CodeTest))的技术资料